Plataforma web de gestión de pacientes en estudios clínicos

Web platform for patient management in clinical trials

Trabajo de Fin de Grado
Curso 2020–2021

Autores
Frederick Ernesto Borges Noronha
y
Carla Paola Peñarrieta Uribe

Director
José Luis Ayala Rodrigo

Tutor
Josué Pagán Ortiz

Trabajo de Fin de Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Plataforma web de gestión de pacientes en estudios clínicos

Web platform for patient management in clinical trials

Grado en Ingeniería Informática

Autores
Frederick Ernesto Borges Noronha
y
Carla Paola Peñarrieta Uribe

Director
José Luis Ayala Rodrigo

Tutor
Josué Pagán Ortiz

Convocatoria: *Junio 2021*

Trabajo de Fin de Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

15 de junio de 2021

Dedicatoria

Este trabajo está dedicado especialmente a todas esas personas que presentan problemas oncológicos, ya que la plataforma que hemos diseñado esta especialmente pensada para ellos.

Además, también queremos dedicar este trabajo a nuestras familias, en especial a nuestros padres, y amigos, que han estado en todo momento apoyándonos de forma incondicional.

"Lo que con mucho trabajo
se obtiene, más se ama."

Aristóteles

Agradecimientos

Queremos agradecer a todas esas personas que nos han ayudado en la elaboración de nuestro Trabajo de Fin de Grado, ojalá pudiésemos nombrarlas a todas con nombre y apellido ya que se lo merecen.

Para empezar, nos gustaría dar las gracias a nuestros tutores José Luis Ayala Rodrigo y Josué Pagán Ortiz, por guiarnos durante estos meses en este trabajo que ha sido un reto para nosotros, siempre aconsejándonos y manteniendo el control sobre todo para que pudiésemos salir adelante.

Queremos dar las gracias a nuestras familias, en especial a nuestros padres, que siempre han estado para dar una mano cuando se necesita, escuchando nuestros problemas, ayudándonos con las revisiones de los textos, preguntándonos si estamos bien luego de estar toda la noche preocupados por este trabajo.

También queremos agradecer a nuestros amigos, que siempre han estado ahí y nos apoyado en todo, teniendo en cuenta que al hacer el trabajo hemos tenido que, indiscutiblemente, decir que no a algunas reuniones, pero no tenían problema ya que sabían que era con un motivo, obtener nuestro tan deseado título universitario.

Por último, pero no menos importante, un especial agradecimiento a Veronika, que nos ha ayudado moralmente y con sugerencias para mejorar nuestro Trabajo de Fin de Grado, con el fin de obtener una mejor calificación.

A todos ustedes,
Muchas gracias.

Resumen

Con este trabajo, pretendemos cambiar un poco el panorama actual de estudios clínicos oncológicos, los cuales requieren una interacción constante por parte del médico y el paciente. Por este motivo hemos decido aportar nuestro esfuerzo para hacer que las visitas constantes al hospital sean reducidas al mínimo.

En este trabajo, se describe la creación de una plataforma web para facilitar la lectura de datos provenientes de dispositivos wearables, suministrados a pacientes oncológicos, para facilitar sus estudios clínicos.

Para realizar esto, se extrajeron los datos de los dispositivos, se procesaron y almacenaron para su posterior uso en dicha web. Para realizar esto hemos dividido cada uno de los componentes necesarios para su elaboración y hemos trabajado en una arquitectura basada en Kubernetes para facilitar su mantenimiento y escalabilidad.

Una vez realizado todo esto, podemos indicar que el resultado final ha sido el esperado para una primera versión de esta web. Hemos permitido que el doctor introduzca los datos de sus pacientes para posteriormente proceder a la obtención de datos con el dispositivo y así poder visualizarlos en la página web.

Palabras clave

- Kubernetes
- Internet de las cosas
- Docker
- Medicina
- Wearables
- Bases de datos
- Procesamiento
- API
- Servicios en la nube

Abstract

With this work, we intend to change the current panorama of oncological clinical studies, which require constant interaction between the doctor and the patient. For this reason, we have decided to contribute our efforts to reduce constant visits to the hospital to a minimum.

This project describes the development of a web platform to make it easier to read data from wearable devices provided to oncology patients to facilitate their clinical studies.

To do this, data was extracted from the devices, processed and stored for subsequent use on the website. In order to achieve this, we have divided each of the components necessary for its development and we have worked on an architecture based on Kubernetes to ease its maintenance and scalability.

Once all this has been done, we can say that the final result has been as expected for a first version of this website. We have allowed the doctor to enter his patients' data and then proceed to obtain the data with the device, so that it can be displayed on the website.

Keywords

- Kubernetes
- Internet of Things
- Docker
- Medicine
- Wearable
- Databases
- Processing
- API
- Cloud Services

Índice

Lista de Figuras

Lista de Tablas

Lista de Acrónimos

ACC Accelerometry

AWS Amazon Web Service

API Application Programming Interfaces

BVP Blood Volume Pulse

CPET Cardiopulmonary Exercise Testing

CSV Comma-separated Values

CoAP Constrained Application Protocol

CGM Continuous Glucose Monitoring

CO_2 Dióxido de carbono

ECG Electrocardiograma

EDA Electrodermal activity

EEG Electroencefalografía

ECV Enfermedad cardiovascular

ENT Enfermedades No Transmisibles

GPS Global Positioning System

HDFS Hadoop Distributed File System

HR Heart Rate

HTTP Hypertext Transfer Protocol

IA Inteligencia Artificial

IoT Internet of Things

IBI Inter-beat interval

JSON JavaScript Object Notation

k8s Kubernetes

ML Machine Learning

NFC Near Field Communication

OMS Organización Mundial de la Salud

RFID Radio Frequency Identification

RNN Recurrent Neural Networks

RHR Resting Heart Rate

SMS Short Message Service

TIC Tecnologías de la información y de la comunicación

TL Telemedicina

TAG Temporal Associative Granulation

UCI Unidades de Cuidados Intensivos

WBSN Wireless Body Sensor Network

WSN Wireless Sensor Networks

Capítulo 1

Introducción

La tecnología *weareable* es un conjunto de dispositivos electrónicos que se incorporan en distintas partes de nuestro cuerpo con el fin de recolectar información, principalmente sobre la salud. Su uso se inició a mediados de la década de los 2000, pero es este año cuando ha alcanzado su punto álgido debido a la epidemia del Covid-19. Gracias a que estos dispositivos nos ofrecen datos sobre el estado fisiológico de los usuarios, se está comprobando que puede llegar a ser una herramienta muy útil sobretodo en el ámbito de la medicina (Roblyer, 2020).

En un reciente estudio (Channa et al., 2020), se ha podido comprobar que estos dispositivos verdaderamente contribuyen en la detección de patologías. Para el estudio utilizaron el dispositivo Fitbit[1], el cual recopilaba los datos del sueño y la frecuencia cardíaca en reposo (RHR, por sus siglas en inglés) y evaluaba si se detectaba alguna anomalía en estos datos que pudiesen ser compatibles con la gripe. Los investigadores concluyeron que este tipo de tecnología puede ayudar de manera positiva en la detección de enfermedades gracias a los datos fisiológicos que se captan a través de los sensores y a que esos datos pueden ser consultados y estudiados por un médico sin necesitar una estrecha relación médico-paciente, que como sabemos y hemos podido comprobar durante la pandemia provocada por el Covid-19, que no es fácil de mantener.

Por otro lado, el uso de la tecnología wearable se está expandiendo en la monitorización de enfermedades crónicas, ya que estas y en concreto, las cardiovasculares, la diabetes tipo 2, el cáncer y las enfermedades respiratorias, representan más del 50 % de todas las muertes en todo el mundo. (Yach et al., 2004)

Por ejemplo, para la monitorización de las arritmias cardiacas, antes de la inclusión de los dispositivos wearables, era necesario un control clínico

[1] `https://www.fitbit.com` Accedido: 17/05/2021

en el que se realizaban las pruebas pertinentes, desde paseos de seis minutos (Mangado y Nieto, 2016) controlando los signos vitales (frecuencia cardíaca, saturación de oxígeno y tensión arterial en reposo) o con la prueba de esfuerzo cardiopulmonar (CPET, de sus siglas en inglés) que solo se realizaba en algunos pacientes debido a su coste elevado (Singhal y Cowie, 2020). Actualmente, muchos dispositivos wearables permiten ampliar la recopilación de datos del paciente al recoger en tiempo real, información necesaria para la monitorización de las arritmias como el ritmo cardiaco, número de pasos y nivel de actividad diaria.

Para el tratamiento de la diabetes, también es muy importante la monitorización de la glucosa (Cappon et al., 2017), al encontrarse en grandes cantidad en la sangre. Existen dispositivos tradicionales que permiten el autocontrol de la glucosa en sangre (SMBG, de sus siglas en inglés) que mediante una punción en el dedo permiten obtener una pequeña gota de sangre para insertarla en un dispositivo que mide el nivel de la glucosa en sangre. Sin embargo, en los últimos años, se han desarrollado *wereables* que incluyen sensores (CGM, de sus siglas en inglés) que pueden medir la concentración de glucosa en sangre o a través del sudor (Gao et al., 2018) casi de forma continua durante las 24 horas del día. Posteriormente, toda esa información podría permitir al médico, regular la cantidad de insulina que necesita cada paciente.

Los dispositivos *wearables*, también están siendo usados para un control de los ataques epilépticos. Existen dispositivos corporales que incluyen sensores capaces de medir la electroencefalografía (EEG), que mide cambios en la actividad cerebral y el electrocardiograma (ECG), que registra la señal eléctrica del corazón. Estos sensores, haciendo uso de algoritmos que analizan las medidas obtenidas, son capaces de predecir los ataques epilépticos minutos antes de que ocurran. Algunos de ellos, también pueden enviar alertas al médico o familiares del paciente a un dispositivo móvil (Mehta et al., 2019).

Otra de las enfermedades en la que interviene la tecnología *wearable* es en la migraña. Al tratarse de un trastorno psicológico, los síntomas más comunes son: dolores de cabeza, mareos, vómitos y sensibilidad a la luz. En muchos casos, no se producen síntomas previos al ataque, por lo que es muy difícil su prevención. Por ese motivo, en algún estudio se ha utilizado una red inalámbrica de sensores corporales (WBSN), que permitía monitorizar las señales fisiológicas del paciente las 24 horas del día. Tras la finalización del estudio, gracias al análisis de la información obtenida mediante modelos predictivos, concluyeron que era posible predecir un ataque de migraña hasta 52 minutos antes de que se produzca (Pagán et al., 2015).

Al igual que para las enfermedades mencionadas previamente, los pacientes oncológicos (Beg et al., 2017), se han visto favorecidos gracias a que al proporcionarles un dispositivo fácil de llevar y lo más importante que mo-

nitorea su actividad han podido mejorar su calidad de vida. Además de un control de los pasos, dispositivos más completos como el que vamos a tratar en este proyecto que miden la sudoración, ritmo cardíaco, aceleración y temperatura han ayudado favorablemente incluso a una detección temprana de la enfermedad.

Como se ha relatado anteriormente, el uso de dispositivos wearables nos aporta infinitos beneficios pero también algunas dificultades ya que, al tratarse de dispositivos que nos proporcionan información en tiempo real, se obtienen grandes ingestas de datos de distintas fuentes que sin un correcto almacenamiento, estructuración, procesamiento, visualización y análisis son inútiles.

El principal problema no es la gran cantidad de datos que nos proporcionan, sino que se basa en que la inclusión de estos dispositivos en el ámbito de la salud es medianamente nuevo, por lo que existen pocos sistemas en los que se permita obtener información de diferentes dispositivos o tecnologías como pueden ser: dispositivos *Internet of Things* (IoT) donde todos esos datos en tiempo real están almacenados en bases de datos de series temporales, en bases de datos relacionales en las que se puede encontrar todo lo relativo a los usuarios, información que proviene de aplicaciones móviles, etcétera. Además, en muchos casos, su propósito no es mostrar esos datos a un usuario final no experto (como el médico o paciente), sino que está orientado a la gestión interna.

Por este motivo, los grupos de investigación interdisciplinares que trabajan en varios proyectos (similares en su arquitectura), orientados a la monitorización de pacientes crónicos, wearables y aplicaciones, no cuentan con una estructura de bloques sólida, lo que dificulta su replicación, mantenimiento y actualización.

Por todo lo anterior, también es importante que todo ese flujo se refleje en una interfaz de manera que el usuario final no experto (generalmente médicos) pueda ver y analizar fácilmente dicha información. El verdadero éxito se centra en conseguir una arquitectura que mantenga una integración eficiente de las bases de datos, el procesamiento, visualización, producción y, que además, sea fácilmente replicable para estudios similares, fácil de actualizar y mantener por personal con conocimientos técnicos. Gracias a esta unificación, conseguiremos que el médico pueda obtener la información del paciente en un mismo sitio.

El desarrollo de este TFG, está incluido en uno de los proyectos de los grupos de investigación GreenDISC UCM y GreenLSI UPM que colaboran con varios hospitales, con el uso de tecnología eHealth en entornos similares como en el caso de migrañas, cefaleas en racimo, y en el que nos vamos a centrar, en oncología. Como ya hemos visto, tener que replicar y mantener sistemas similares pero diferentes según sus ámbitos y necesidades es difícil, por eso se plantean los siguientes objetivos.

1.1 Objetivos

La finalidad de nuestro proyecto se basa en conseguir una arquitectura sólida que facilite la integración de nuevas fuentes de datos y aplicaciones en un mismo entorno y, además que no solo sirva para el ámbito de la salud, sino sea aplicable a otras disciplinas. Los datos con los que vamos a tratar no son solo los que tienen que ver con la información personal sino que también contamos con los datos que nos proporciona una pulsera de monitorización biométrica (E4 *wristband*)[2] que es usada por pacientes oncológicos y ha sido proporcionada por nuestros tutores.

Una vez dicho esto, para conseguir el objetivo, los dividiremos en principales y secundarios.

1.1.1 Objetivos principales

1. **Definición de la arquitectura**: se diseñará una estructura estándar en Docker haciendo uso de Kubernetes para la integración de datos de series temporales, datos relacionales de seguimiento, integración de módulos de procesamiento de datos y un módulo de web.
2. **Implementación**: la arquitectura se implementará en un entorno real Cloud para facilitar el acceso a los datos.
3. **Despliegue**: el despliegue se realizará para un caso concreto, en base a una necesidad del equipo de investigación, la monitorización de pacientes con dolor oncológico.

1.1.2 Objetivos secundarios

1. Definición y diseño de una plataforma web de prueba para médicos.
2. Inclusión del "scrapper" como otro módulo en Kubernetes, que permitirá la obtención de datos de la pulsera de monitorización ambulatoria *E4 wristband*.
3. Integración de mecanismos para el procesamiento de los datos los datos biométricos.
4. Incluir una interfaz para permitir a los pacientes visualizar su información personal.

1.2 Estructura del TFG

En los siguientes capítulos se expondrá el Estado del Arte (capítulo 2) en el que conoceremos sobre diferentes estudios en los que se proponen soluciones

[2] `https://www.empatica.com/en-eu/research/e4/` Accedido: 17/05/2021

alternativas al principal problema a abordar, la Metodología del Software (capítulo 3), donde se podrá ver el modelo y el flujo de trabajo que mejor se adaptaba a nuestro proyecto, las Herramientas (capítulo 4) utilizadas para la implementación del TFG y que han ido formando parte del proceso de investigación, la Implementación (capítulo 5) donde vendrán explicados las diferentes aproximaciones que se han realizado sobre la arquitectura hasta la finalmente obtenida junto con todo su desarrollo, en los Resultados (capítulo 6) podremos visualizar lo finalmente conseguido tras lo que hemos ido viendo en los capítulos anteriores, el Trabajo individual (capítulo 7) de cada uno de los integrantes del proyecto, donde se describen las tareas realizadas por cada uno de ellos, en el Trabajo Futuro (capítulo 8) se describirán los aspectos a mejorar así como nuevas ideas para futuras versiones y por último, las Conclusiones (capítulo 9) en las que vendrá reflejado lo que se ha obtenido con la realización de este proyecto y hasta que punto se han conseguido los objetivos planteados al inicio.

Chapter 1

Introduction

Wearable technology is a set of electronic devices incorporated into different parts of our body to collect information, mainly about our health. Its use began in the mid-2000s, but it reached its peak this year due to the Covid-19 epidemic. Thanks to the fact that these devices provide us with data on the physiological state of users, it is proving to be a helpful tool, especially in the field of medicine.

In a recent study (Channa et al., 2020), it has been proven that these devices really contribute to the detection of diseases. For the study they used the Fitbit[1] device, which collected information on their sleep and resting heart rate (RHR), and evaluated whether any anomalies were detected in this data that could be compatible with a flu-like illness. The researchers concluded that this type of technology can help positively in the detection of disease thanks to the data captured by the sensors, and the fact that these data can be consulted and studied by a doctor without the need for a face-to-face doctor-patient interaction, which, as we know and as we have seen during the Covid-19 pandemic, is not always easy to maintain.

On the other hand, the use of wearable technology is expanding in the monitoring of chronic diseases, as these, specifically cardiovascular diseases, type 2 diabetes, cancer and respiratory diseases, account for more than 50

For example, monitoring for cardiac arrhythmias, prior to the inclusion of wearable devices, required clinical management where relevant tests were performed, from six-minute walks monitoring vital signs (heart rate, oxygen saturation and blood pressure at rest) or the cardiopulmonary stress test (CPET), which was only performed on some patients due to its high cost. Currently, many wearable devices allow expanded patient data collection by collecting, in real time, information needed for arrhythmia monitoring such as heart rate, number of steps and daily activity level.

[1] `https://www.fitbit.com`

For the treatment of diabetes, glucose monitoring is also very important, as it is found in large quantities in the blood. There are traditional self-monitoring blood glucose (SMBG) devices that allow a small drop of blood to be obtained by pricking the finger and inserted into a device that measures the level of glucose in the blood. However, in recent years, *wereables* have been developed that include sensors (CGM) that can measure glucose concentration in blood or through sweat almost continuously 24 hours a day. Subsequently, all this information could allow the physician to regulate the amount of insulin needed for each patient.

Wearable devices are also being used to monitor epileptic seizures. There are body-worn devices that include sensors capable of measuring electroencephalography (EEG), which measures changes in brain activity, and electrocardiogram (ECG), which records the heart's electrical signal. These sensors, using algorithms that analyze the measurements obtained, are able to predict epileptic seizures minutes before they occur. Some of them can also send alerts to the patient's physician or family members to a mobile device.

Another disease in which wearable technology is involved is migraine. As a psychological disorder, the most common symptoms are headaches, dizziness, vomiting and sensitivity to light. In many cases, there are no symptoms prior to the attack, making it very difficult to prevent. For this reason, a wireless body sensor network (WBSN) has been used in some studies to monitor the patient's physiological signals 24 hours a day. After the completion of the study, thanks to the analysis of the information obtained through predictive models, they concluded that it was possible to predict a migraine attack up to 52 minutes before it occurs.

As with the previously mentioned diseases, oncology patients have been helped by the fact that by providing them with a device that is easy to wear and most importantly monitors their activity, they have been able to improve their quality of life. In addition to step monitoring, more comprehensive devices such as the one we will discuss in this project that measure sweating, heart rate, acceleration and temperature have even favorably aided in early detection of the disease.

As mentioned above, the use of wearable devices brings us infinite benefits but also some difficulties since, being devices that provide us with information in real time, large amounts of data are obtained from different sources that are useless without proper storage, structuring, processing, visualization and analysis.

The main problem is not the large amount of data they provide us with, but is based on the fact that the inclusion of these devices in the field of healthcare is fairly new, so there are few systems in which it is possible to obtain information from different devices or technologies such as: Internet of Things (IoT) devices where all these real-time data are stored in time series databases, in relational databases in which everything related to the users

can be found, information from mobile applications, etc. Moreover, in many cases, their purpose is not to show these data to a non-expert end user (such as the doctor or patient), but is oriented to internal management.

For this reason, interdisciplinary research groups working on several projects (similar in their architecture), oriented to chronic patient monitoring, wearables and applications, do not have a solid block structure, which makes their replication, maintenance and updating difficult.

For all of the above, it is also important that all this flow is reflected in an interface so that the non-expert end user (usually physicians) can easily view and analyze this information. The real success lies in achieving an architecture that maintains an efficient integration of databases, processing, visualization, production and, in addition, is easily replicable for similar studies, easy to update and maintain by personnel with technical knowledge. Thanks to this unification, the physician will be able to obtain patient information in the same place.

The development of this TFG is included in one of the projects of the research groups GreenDISC UCM and GreenLSI UPM that collaborate with several hospitals, with the use of eHealth technology in similar environments as in the case of migraines, cluster headaches, and the one we are going to focus on, oncology. As we have already seen, having to replicate and maintain similar but different systems according to their fields and needs is difficult, so the following objectives are proposed.

1.1 Goals

The purpose of our project is based on achieving a solid architecture that facilitates the integration of new data sources and applications in the same environment and, moreover, that not only serves for the field of health, but is applicable to other disciplines. The data we are going to deal with are not only those that have to do with personal information but we also have the data provided by a biometric monitoring bracelet (E4 *wristband*)[2] that is used by oncology patients and has been provided by our tutors.

Una vez dicho esto, para conseguir el objetivo, los dividiremos en principales y secundarios.

1.1.1 Primary objectives

1. **Architecture definition**: a standard Docker framework will be designed using Kubernetes for the integration of time series data, relational data tracking, integration of data processing modules and a web module.

[2] `https://www.empatica.com/en-eu/research/e4/` Accessed: 17/05/2021

2. **Implementation**: the architecture will be implemented in a real Cloud environment to facilitate data access.

3. **Deployment**: the deployment will be carried out for a specific case, based on a need of the research team, the monitoring of patients with oncological pain.

1.1.2 Secondary objectives

1. Definition and design of a web-based testing platform for physicians.

2. Inclusion of the "scrapper" as another module in Kubernetes, which will allow obtaining data from the *E4 wristband* ambulatory monitoring bracelet.

3. Integration of mechanisms for processing biometric data.

4. Include an interface to allow patients to view their personal information.

1.2 Document structure

The following chapters will present the State of the Art (chapter 2) in which we will learn about different studies in which alternative solutions to the main problem to be addressed are proposed, the Software Methodology (chapter 3), where we will see the methodology and the workflow that best suits our project, the Tools (chapter 4) used for the implementation of the final project and that have been part of the research process, the Implementation (chapter 5) where the different approaches that have been taken on the architecture up to the final one, as well as its development will be explained, in Results (chapter 6) we will be able to visualize what has finally been achieved after what we have seen in the previous chapters, the Individual Work (chapter 7) of each of the members of the project, where the tasks realized by each one of them are described. In Future Work (chapter 8) we will describe the aspects to be improved as well as new ideas for future versions and finally, the Conclusions (chapter 9) in which it will be reflected what has been obtained with the implementation of this project and to what extent the objectives established at the beginning have been achieved.

Capítulo 2

Estado del arte

Hemos dividido este capítulo en tres bloques en el primero se explicarán que son los dispositivos IoT y para que se usan (sección 2.1). El segundo bloque, se centrará en explicar el concepto de *Smart Health* (capítulo 2.2) y se presentarán algunos ejemplos de plataformas o implementaciones utilizadas en este entorno (sección 2.2.1). Por último, en el tercer bloque se explicará el concepto de Recolección de datos (sección 2.3), su importancia y lo que nos ha aportado en la realización de este proyecto.

2.1 Internet de las cosas

Gracias a que la tecnología está en continuo desarrollo, tenemos más herramientas que nos ayudan a tener una mejor calidad de vida. Un claro ejemplo de esto es el progreso de la tecnología Internet of Things[1] (IoT). Se trata de dispositivos que están integrados con sensores, software y otras tecnologías con la finalidad de conectar e intercambiar datos con otros dispositivos a través de conexiones inalámbricas. Este tipo de tecnología, se ha convertido en algo imprescindible en nuestra sociedad por lo que su uso se ha expandido en diferentes ámbitos: educación, trabajo, agricultura y ganadería, logística, negocios, salud, entre los más importantes.

El sector de la salud es uno de los que más interesa a los investigadores, sobre todo después de lo acontecido este año con la pandemia mundial (Covid-19). Gracias a eso, son cada vez más los médicos que proporcionan diferentes dispositivos a sus pacientes, como pueden ser para el control de la glucosa, del ritmo cardiaco, para el control de pacientes oncológicos, con Parkinson o alguna otra enfermedad crónica e incluso dispositivos que monitorizan el estado de ánimo.

[1] `https://www.oracle.com/es/internet-of-things/what-is-iot/` Accedido: 17/05/2021

Como ya hemos comentado, cada sector tiene un propósito en común, que es el encontrar una arquitectura estándar que permita gestionar de manera eficiente la ingesta de datos que provienen de diferentes fuentes. En este capítulo, nos vamos a centrar en el sector de la medicina y en el concepto de *Smart Health*.

2.2 Smart Health

Durante los últimos años se ha podido ver que la industria de la salud está muy debilitada y en mucho casos, no cuentan con el personal y los recursos necesarios para tratar de forma más personalizada a cada paciente. El hecho de que un médico tenga que desgastarse física y mentalmente puede provocar situaciones de estrés y como consecuencia, verse reflejado en el tratamiento de los pacientes.

A lo largo de los años, la situación sanitaria puede empeorar ya que según la Organización Mundial de la Salud (OMS)[2], "Las enfermedades cardiovasculares constituyen la mayoría de las muertes por enfermedades no transmisibles (ENT) (17,9 millones cada año), seguidas del cáncer (9,0 millones), enfermedades respiratorias (3,9 millones) y diabetes (1,6 millones)".

El concepto de *Smart Health* está sirviendo de respaldo en la industria sanitaria. Este último, se trata de una combinación entre el conjunto de TICs que se emplean en el sector sanitario (*eHealth*) y el uso de la tecnología móvil en beneficio de la salud *(mHealth*).

Gracias a la combinación de *eHealth* y *mHealth*, el personal sanitario puede ofrecer mejores diagnósticos y control, en tiempo real, de la información que los dispositivos transmiten. Además, ha ayudado a que la idea de *Smart Hospital* (hospitales inteligentes) sea aplicado a más hospitales, ya que tiene la finalidad de realizar el seguimiento de los pacientes a través de las TICs, sin necesidad de trasladarse a un hospital, ya que la información recopilada por los dispositivos puede ser consultada por el médico remotamente. Esto permite que se puedan detectar emergencias a mayor velocidad, evitando en muchos casos, el ingreso en los hospitales. Como prueba de ello, en un programa de *Partners HealthCare* (Kvedar et al., 2014), más de 3.000 pacientes fueron controlados desde sus casas haciendo uso de tecnología que medía la presión arterial, la frecuencia cardíaca y oximetría del pulso. Los datos eran procesados por un software que identificaba a los pacientes que necesitaban atención médica. Estos datos eran revisados por enfermeras en grandes paneles, por lo que no solo redujeron el costes (10 millones de dolares durante 6 años) sino que como resultado del estudio se redujeron un 44 % los reingresos en los hospitales.

Para poner conocer como se aplican estos dos conceptos, comenzaremos

[2] `https://www.who.int/es/news-room/fact-sheets/detail/noncommunicable-diseases` Accedido: 20/05/2021

dando un repaso por algunos de los dispositivos IoT utilizados en la monitorización de enfermedades y cómo se realiza el seguimiento en remoto de paciente. Además, podremos ver ejemplos de plataformas o implementaciones en la que se hecho uso de este tipo de tecnología, destacando las ventajas y desventajas de cada una de ellas.

2.2.1 Dispositivos IoT y entornos de monitorización

Es importante conocer como ha evolucionado la comunicación médico-paciente a lo que se conoce como Telemedicina (TL), que se define como cualquier actividad médica realizada a distancia (Wootton, 2001). Esto no solo ha beneficiado a la comunicación de médicos y pacientes, sino que ha permitido que se puedan monitorizar enfermedades de forma remota gracias al uso de la tecnología IoT. Por eso, a lo largo de esta sección veremos como se han incluido estos dispositivos en los entornos de monitorización.

Por ejemplo, en un estudio, se ha realizado el seguimiento de algunos pacientes que padecen apnea del sueño (Sannino et al., 2014). Antiguamente, las personas que lo sufrían, eran sometidas a una prueba muy compleja y costosa que requería asistencia presencial. En este caso, para la monitorización de la apnea se propone el uso de *The BioHarness BH3* (Figura 2.1).

Se trata de un dispositivo en forma de cinturón, que monitorea datos fisiológicos y que usa Bluetooth para transmitir datos. Es capaz de medir el ECG, la frecuencia cardíaca, frecuencia respiratoria y temperatura y un acelerómetro de tres ejes para el control de la postura y actividad.

Figura 2.1: The BioHarness BH3
Fuente: Amazon

Mientras el paciente duerme, el sensor incluido en el cinturón envía datos en tiempo real a un dispositivo móvil que, posteriormente, calcula y obtiene reglas que permiten saber si el paciente está sufriendo de apnea del sueño.

Cada una de las reglas son únicas para cada paciente que, en caso de variaciones en los datos, se generan alarmas para avisar al paciente o médico. Además, el sistema de monitoreo, a pesar de haber usado solo datos del ECG, la extracción datos y el algoritmo propuesto para la obtención de las reglas permite incluir información de otros datos fisiológicos pero solo si provienen del mismo dispositivo. Finalmente, a través de una interfaz de usuario, se puede visualizar los resultados obtenidos, como se puede ver en la Figura 2.2.

Figura 2.2: Interfaz de usuario

En algunos estudios, han incluido otro tipo de tecnología IoT, como son los dispositivos implantables. Como se puede ver en la Figura 2.3, *Confirm Rx*, ha sido diseñado por Abbott[3], una empresa que realiza investigaciones y fabrica productos para salud humana. Su dispositivo es subcutáneamente implantado y permite medir el ritmo cardiaco de forma continua, por lo que está formado por un sistema que detecta arritmias cardiacas. Después de la incorporación del software *SharpSense™*, se consiguió eliminar las falsos positivos, manteniendo su capacidad de detección. Además, este permite conexión vía Bluetooth con la aplicación móvil *exmyMerlin™*, a la que envía los datos continuamente, alarmas al paciente y médico, permite anotar los síntomas y, además, envía la información a *myMerlin™ Patient Care Network*, una interfaz web que permite a los médicos cargar y administrar los datos únicamente del dispositivo *Confirm Rx* de cada paciente.

En uno de esos estudios, se probó la eficacia de este dispositivo en pacientes de entre 11 meses y 21 años. Se les implantó a cada uno de ellos uno de estos dispositivos con los que eran monitorizados como comentamos

[3] `https://www.es.abbott/` Accedido: 09/06/2021

Figura 2.3: Dispositivo implantable Confirm Rx™
Fuente: Geriatricarea

anteriormente. De manera que, cada paciente podría ser controlado a través de una interfaz web, por su pediatra y además, alertado en caso de arritmias o síntomas. Finalmente, concluyeron que es eficaz para la detección de arritmias y seguro de usar en pacientes pediátricos (Yoon et al., 2020).

Para la monitorización de la diabetes también existen dispositivos que ayudan a tener mayor control de la glucosa. Aunque es algo medianamente nuevo, se ha visto que su uso puede resultar muy beneficioso, tanto en adultos como en niños (Larson y Pinsker, 2013).

En la Figura 2.4, podemos observar un ejemplo de dispositivo que permite la monitorización continua de la glucosa, *FreeStyle Libre*, fabricado por Abbott. Cuenta con un sensor que se coloca en el brazo o estómago, el cual manda información a este dispositivo. Este mismo, permite la visualización de gráficas donde viene indicado el valor de la glucosa en sangre por horas. Además, tiene conexión vía Bluetooth con una aplicación móvil dónde se puede acceder a más información.

Figura 2.4: Dispositivo FreeStyle y sensor
Fuente: El diario Montañés

Otro factor importante en el control de la glucosa es el ejercicio. Existen

una infinidad de dispositivos que miden la actividad física, como por ejemplo la pulsera *Fitbit* (Figura 2.5). Al igual que *FreeStyle Libre*, cuenta con una aplicación móvil donde se pueden ver en forma de gráficas un histórico de los pasos, del sueño, calorías quemadas, etcétera. Los más modernos también son capaces de medir datos fisiológicos, como en este caso.

Figura 2.5: Pulsera Fitbit
Fuente: Fitbit

FreeStyle Libre y *Fitbit* fueron utilizados en un estudio para la monitorización de la glucosa en pacientes con un riesgo moderado o alto de la Diabetes tipo 2 (Whelan et al., 2019). Participaron personas adultas mayores de 40 años. Se observó que muchos de ellos pudieron cambiar su actividad física gracias a *Fitbit* y por otro lado, mejorar el control de la glucosa con el uso de *FreeStyle Libre*. Aunque en general, se obtuvieron buenos resultado, algunos de los participantes en el estudio tuvieron dificultades, por ejemplo, en cuanto a la durabilidad de los sensores, pues no se mantenían adheridos a la piel más de dos semanas y también, con la visualización de los datos en las aplicaciones móviles, pues en muchos casos no accedían, lo que indicaba que no tenían conocimiento de la información que los sensores captaban. Además, resaltan la necesidad de integrar en una sola plataforma la información recogida por los sensores, ya que los dos dispositivos tenían una aplicación móvil cada uno. También, proponen que se facilite la automatización y retroalimentación de los datos mediante el uso de múltiple fuentes de datos en tiempo real.

Como hemos podido ver, el uso de los dispositivos en entornos de monitorización proporciona muchos beneficios y sobretodo, permite recopilar información remotamente y en tiempo real. En muchos estudios, la información solo es visible a través de la aplicación móvil (si la tuviese) a la que se conecta el dispositivo, siendo accesible únicamente por el paciente, dificultando el acceso a los médicos. En otros casos, el no tener una plataforma

que integre la información de varios dispositivos o sea visible para el médico, dificulta su seguimiento, como hemos podido observar anteriormente.

Por todo lo dicho, el desarrollo de este TFG puede ser de mucha utilidad, ya que supondrá plantear una estructura que permita solucionar muchos de los problemas con los que se han encontrado en algunos estudios de monitorización.

Por eso, en la siguiente sección podremos conocer plataformas o implementaciones en los que se ha propuesto o solucionado alguno de estos problemas.

2.2.2 Plataformas e implementaciones en SmartHealth

En esta sección se presentarán algunos estudios en los que se han desarrollado o propuesto plataformas o implementaciones IoT en el ámbito de *SmartHealth*. Evaluaremos aspectos como su arquitectura, tecnologías aplicadas, interfaz de usuario, procesamiento de datos, etcétera.

Recientemente, en un estudio se ha propuesto un sistema de asistencia sanitaria basada en IoT para pacientes con Alzheimer (Jamili oskouei et al., 2020). La finalidad de este proyecto, se basa en ayudar a monitorear las actividades, datos fisiológicos, y biológicos de las personas que padecen esta enfermedad. Para ello, diseñaron la arquitectura de la Figura 2.6.

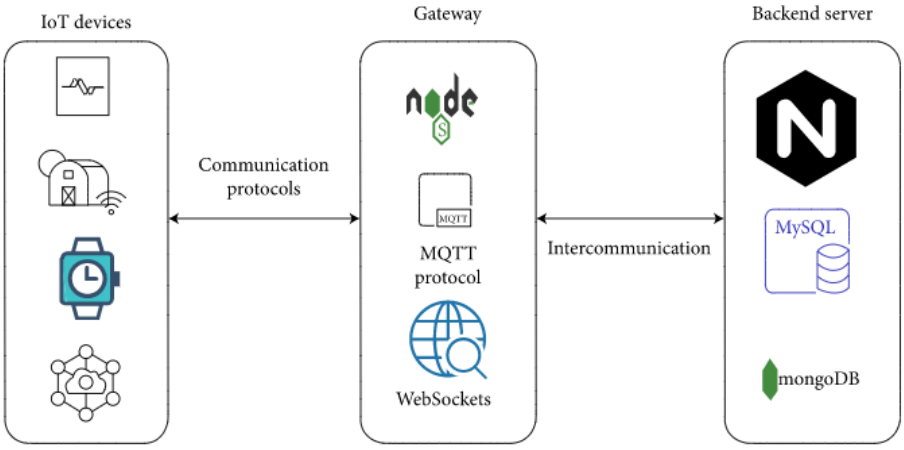

Figura 2.6: Arquitectura para la monitorización del Alzheimer

Dicha arquitectura se puede dividir en tres módulos:

- **Dispositivos IoT**: este tipo de estructura facilita la integración de varios sensores IoT ya que utilizan mecanismos de comunicación que permiten la conexión entre ellos y además, que envíen información al siguiente nivel. Estos mecanismos son el protocolo de transferencia de hipertexto (HTTP), Message Queue Server (MQTT) y WebSockets,

los cuales permiten la comunicación cliente/servidor.

- **Puerta de enlace**: para permitir la comunicación entre dispositivos se desarrollan métodos para cada uno de los protocolos en NodeJS que, en función del tipo de dispositivo, se selecciona uno o otro. Además, el método de conexión que se utiliza para WebSocket, permite que se puedan registrar los dispositivos en la base de datos y alertar en caso de que se registren falsos. El método usado para HTTP, permite que se pueda localizar a dispositivos registrados. De esta manera, sabiendo dónde se encuentra cada paciente se facilitaría su localización en caso de emergencia.

- **Servidor backend**: se utiliza Nginx como proxy inverso para los métodos NodeJS ya que mantiene la privacidad de la información de los usuarios. Por otro lado, se usa MySQL para almacenar los datos del usuario que podría ser usado en proyectos futuros para el análisis de enfermedades. En cuanto al almacenamiento de la información de los dispositivos IoT, se ha usado mongoDB, ya que permite el acceso a la información en tiempo real. Por último, han optado por un almacenamiento en la nube, garantizando la seguridad de los datos, acceso y reduciendo costes.

En relación a la interfaz web, actualmente en estado de prueba, resulta muy útil e intuitiva para los familiares y médicos de los pacientes. Aunque no incluye información personal o sobre el estado de la enfermedad, han incluido la base de datos MySQL en la arquitectura (sin implementar) para solventarlo.

Por ejemplo, en la Figura 2.7 se pueden localizar a los dispositivos registrados, es decir, la localización de cada paciente.

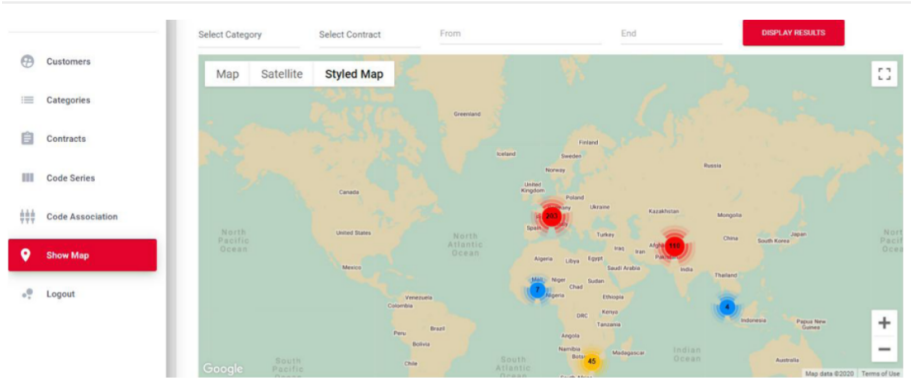

Figura 2.7: Localización de dispositivos

En la Figura 2.8, se pueden ver alertas generadas en base a los datos de

los sensores. Gracias a esto, el médico o familiares pueden actuar en caso de urgencia.

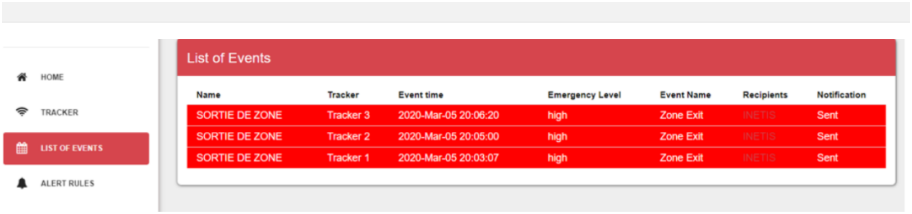

Figura 2.8: Lista de alertas

También incluyen un listado de los dispositivos (Figura 2.9), en el que se muestra información que puede ser importante como el estado de la batería, si está activo o no, número de dispositivo, etcétera.

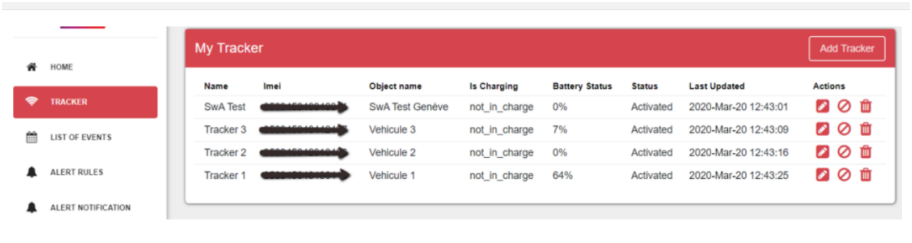

Figura 2.9: Lista de dispositivos

En otro estudio, se propuso una arquitectura y un prototipo de aplicación móvil para monitorizar la salud en personas mayores, todo ello recopilando información de un pulsera biométrica (Durán-Vega et al., 2019). La aplicación móvil llamada Abuelómetro, en este caso, no fue diseñada para ser utilizada por médicos, sino por los cuidadores. Gracias a esto, estos podrían tener acceso a datos generados por los sensores como la frecuencia cardiaca, temperatura corporal y oxigenación sanguínea.

La arquitectura propuesta se puede ver en la Figura 2.10.

Está formada por cuatro capas. En la primera, se encuentra la pulsera biométrica usada en este estudio. Como una de las ventajas, este dispositivo cuenta con software y hardware de código abierto, es decir, resulta sencillo modificarlo en función de la finalidad de cada proyecto. Como desventaja, este prototipo admite únicamente como fuente de datos la del dispositivo usado en este proyecto, no permite cambiar ni incluir otro tipo de tecnología IoT. En la segunda capa, la pulsera biométrica se conecta con una plataforma llamada *WolkAbout IoT Platform*, que cuenta con una aplicación móvil al cual se envía toda la información que captan los sensores y posteriormente, es almacenada en la nube. Gracias a que dicha aplicación cuenta con una API, se pueden acceder a esos datos. En la tercera capa, se desarrolla un

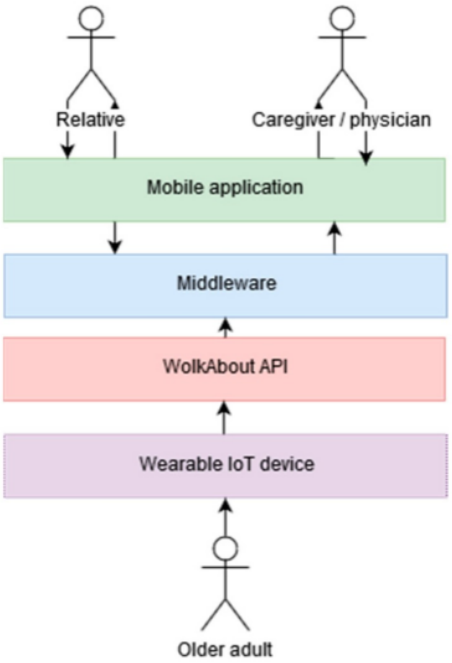

Figura 2.10: Arquitectura del sistema Abuelómetro

Middleware[4], que permitirá la "canalización" de los datos desde la API a la base de datos utilizada, MongoDB (base de datos orientado a documentos). Para el desarrollo del *middleware* se usó NodeJS, ya que ofrece un mejor rendimiento, también porque según algunos estudios (Chaniotis et al., 2015) era mejor opción para el desarrollo de aplicaciones que otras tecnologías como PHP o Apache y, junto a MongoDB, facilitan la consulta de los datos en tiempo real. Finalmente, se ha propuesto un prototipo de la aplicación móvil, en la que los cuidadores podrían iniciar sesión y acceder al historial médico (mediante un calendario) y a una vista donde puede ver los datos obtenidos de los sensores (Figura 2.11), a un chat donde podrían mantener contacto con los familiares y a un sistema de alarmas por colores (Figura 2.12), en el que mediante un algoritmo basado en reglas, permitía detectar anomalías en la información.

En cuanto a usabilidad de la aplicación, se obtuvieron buenos resultados, por lo que se propone su evaluación en residencias geriátricas. Aunque sea haya diseñado únicamente para el caso de personas mayores, manteniendo como fuente de datos la de la pulsera de este estudio, puede ser replicable a otro tipo de proyectos. Como desventaja, resultaría complicado incluir o

[4] `https://azure.microsoft.com/es-es/overview/what-is-middleware/` Accedido: 09/06/2021

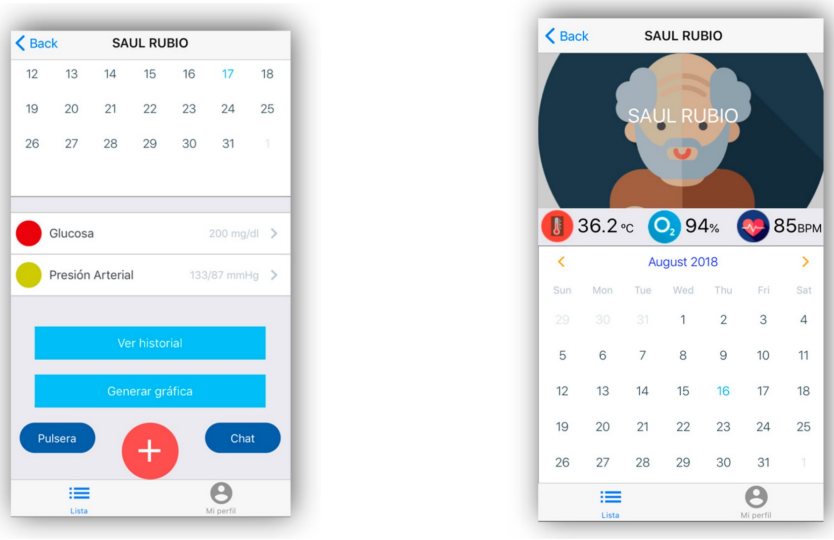

Figura 2.11: Historial médico y lectura del sensor

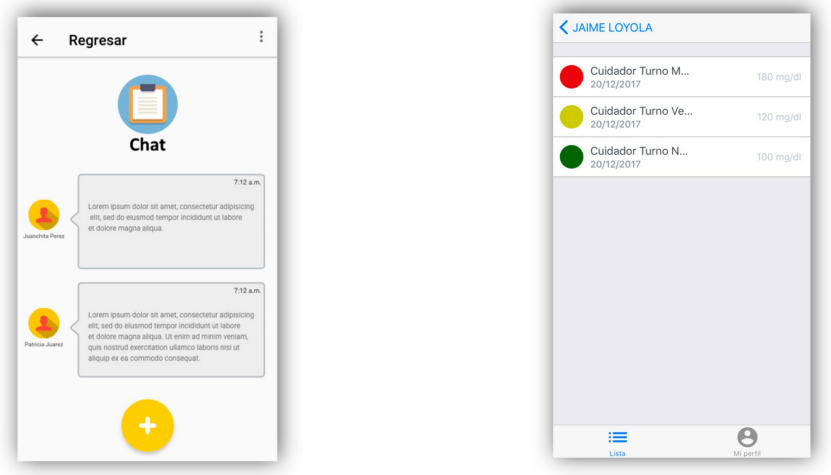

Figura 2.12: Chat cuidador-familiar y lista de alertas

prescindir dispositivos IoT en la arquitectura propuesta o incluir módulos de procesamiento o maching learning en el caso de se opten por dispositivos diferentes al de la pulsera biométrica. Además, se incluye como trabajo futuro, mejorar la seguridad de los datos para evitar cualquier tipo de ataque.

En otro estudio, se realizó un seguimiento de ECG en un entorno *Cloud* en el que los datos no eran enviados a un móvil mediante Bluetooth o Zigbee, sino que se enviaban a un nodo de monitoreo portátil el cual enviaba los datos directamente a la nube mediante WiFi (Yang et al., 2016).

En la Figura 2.13 se representa la arquitectura formada por tres capas:

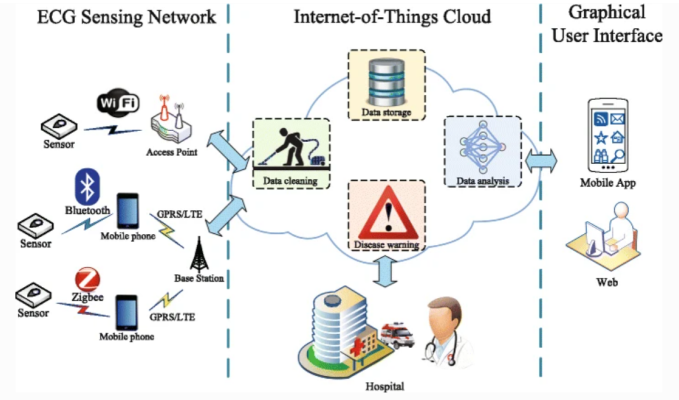

Figura 2.13: Arquitectura de un sistema de monitereo en la nube

- **Red de sensores**: se encarga de recopilar los datos obtenidos por los sensores ECG portátiles basados en *Wireless sensor networks* (WSN) y enviarlos mediante WiFi a la nube. Al utilizar este último como mecanismo de conexión se garantiza la velocidad y una mayor cobertura. Como se puede observar, con el uso de Bluetooth y de Zigbee se necesita tener un móvil como intermediario en la trasmisión de los datos a la siguiente capa.

- **IoT cloud**: esta capa es una de las más importantes ya que los datos serán preparados para su visualización. Se hará uso del protocolo HTTP para la transferencia de recursos a través de Internet.

  En la primera etapa *(Data cleaning)*, se deberá reducir el ruido de la señal ECG para obtener una mejor precisión en los resultados. Para ello, se aplica un filtro que elimina el ruido que esté fuera de la señal ECG.

  Una vez los datos estén limpios, serán almacenados en una base de datos no relacional ya que será necesario guardar una marca de tiempo y el valor de la señal *(Data storage)*. La base de datos elegida en este estudio es Redis[5], ya que se necesita que los datos sean consultados en tiempo real y a la máxima velocidad al menor coste.

  En la última etapa, se analizarán los datos haciendo uso de las herramientas que proporciona un entorno Cloud para ello o incluyendo ML para un mejor diagnóstico o en el mejor de los casos conseguir identificar y diferenciar entre enfermedades cardiacas *(Data Analysis)*. Por otro lado, un factor importante es el aviso al hospital en caso de verdadera emergencia, es decir, en el caso de observar anomalías en los datos en tiempo real se mandaría un aviso de alerta al hospital *(Disease warning)*.

[5] `https://redis.io/` Accedido 08/06/2021

- **Interfaz gráfica del usuario**: mediante una aplicación móvil o plataforma web los datos podrán ser visualizados en tiempo real por el médico.

A lo largo de este capítulo, hemos mencionado en numerosas ocasiones a los dispositivos IoT. En casi todos ellos, los datos generados por sensores se obtenían a través de un dispositivo móvil mediante protocolos como Bluetooth y Zigbee en su mayoría. Pero existen otros mecanismos que permiten obtener automáticamente esa información como el *web scraping* que será explicado a continuación.

2.3 Recolección de datos

Como ya hemos visto, existen diferentes tecnologías que nos permiten enviar y recibir datos a la nube. Pero también existen técnicas que permiten la extracción de información desde cualquier página web de manera automática. Esto se conoce como web scraping.

De trabajo previo del grupo de investigación, se ha integrado una serie de ficheros en los que se utilizaba esta técnica. Están escritos en Python y se encargan de la descarga de las "sesiones" de los usuarios que se encuentran alojadas en la web de Empatica[6]. Se le llama saysesión al paquete .zip que se genera tras encender y apagar la pulsera. Gracias a esta descarga automática, podemos tener acceso a los ficheros que contiene cada sesión con el valor de cada uno de los datos fisiológicos que mide la pulsera que hemos visto anteriormente.

Como hemos podido ver en los diferentes estudios presentados, la finalidad de cada uno de ellos se centraba sobretodo, en proponer una solución al problema que querían abordar en cada caso en concreto. Como consecuencia de ello, se han propuesto diferentes arquitecturas y tecnologías que, en muchos casos, dificultarían la inclusión de dispositivos nuevos o prescindir de ellos y poner reutilizar el resto de bloques, módulos de procesamiento o incluso añadir un bloque de maching learning (ML) para realizar predicciones en base a los datos.

Finalmente, viendo el estado del arte, se propondrá una estructura estándar para que pueda ser replicable o modificable a otros proyectos y además, en base a las tecnologías, técnicas, entornos y métodos de almacenamiento usados en muchos estudios, se optará por las que nos permitan la unificación de cada uno de los bloques de los que estará formada nuestra plataforma.

[6] `https://www.empatica.com/connect/login.php` Accedido: 08/06/2021

Capítulo 3

Metodología del Software

La metodología del software es utilizada al principio del proyecto para llevar a cabo una planificación, permitiendo de esta manera que aquellas personas que se encargarán de llevar a cabo dicho trabajo, sean conscientes de todas aquellas cosas que se deben realizar para así obtener el resultado final esperado. Los dos tipos de metodologías del software más comunes son: las tradicionales y las ágiles.

En las metodologías tradicionales el flujo de proceso se lleva a cabo de manera lineal, siendo difícil aplicar modificaciones, pero son utilizadas principalmente cuando los requerimientos están muy bien establecidos, es decir se conoce cual es el trabajo que se va a realizar y además se sabe cuanto tiempo se tiene para aplicarlos. Por otra parte, en las metodologías ágiles el flujo de proceso es iterativo y se puede pensar con mayor facilidad en incluir modificaciones para mejorar el desarrollo del software, este modelo se utiliza cuando se esperan muchos cambios pero tiene como inconveniente que requiere la presencia constante del cliente.

Para este Trabajo Final de Grado hemos optado por utilizar una metodología tradicional llamada *Modelo de la Cascada* debido a varios aspectos que destacamos a continuación:

- Se tenían dos sistemas ya existentes, los cuales querían mejorarse permitiendo una conexión entre ambos.
- Los requerimientos para dicha conexión fueron especificados desde el inicio.
- El proyecto poseía estabilidad, lo que quiere decir que no sufre cambios de último momento o similares.
- Establecida las necesidades y partiendo del material previo existente, las especificaciones pertinentes son conocidas desde el inicio.

- La definición de un sistema modular que alberga elementos intercambiables hace que sea sencillo incluir elementos ya existentes o cambiarlos por especificaciones nuevas sin alterar la estabilidad del sistema.

Con los puntos descritos anteriormente, se hizo sencillo adoptar el *Modelo de la Cascada*, ya que este modelo según el libro "*Ingeniería del software. Un enfoque práctico*" (Pressman, 2010), se puede adoptar "cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente"

3.1 Modelo de la cascada

Este modelo de desarrollo de software permite definir las fases en las que se van a trabajar con antelación. Cada una de las actividades de estas fases debe ser realizada y entregada antes de comenzar con la siguiente etapa. La idea general es que se tengan al menos fases para definir los requerimientos del software, cómo se realizará el proyecto, codificar los cambios que se van a realizar para obtener el resultado esperado, pruebas sobre los resultados obtenidos y por ultimo la implementación, la cual se centra en tareas como el despliegue del proyecto realizado y además la redacción de una buena documentación.

La aplicación de este modelo de metodología del software viene dada ya que, o bien los requerimientos de aquello que se desea realizar son bien comprendidos desde un inicio, o bien el trabajo a realizar son solo mejoras sobre un proyecto. Este esquema de trabajo presenta un flujo de trabajo secuencial en el se diferencian bien las etapas por las que se pasa y además es muy sencillo de aplicar. En la figura 3.1 se puede observar el modelo de cascada utilizado para este proyecto, donde se pueden observar las siguientes fases:

Figura 3.1: Esquema del modelo de la cascada implementado.

- **Comunicación**: En esta fase se habla con los clientes, en nuestro casos los tutores del proyecto, para determinar los requisitos del sistema para luego poder realizar la etapa de investigación.

- **Investigación**: En esta etapa se procede a indagar sobre los conceptos que estarán presentes en el proyecto para así tener una idea clara y poder empezar la etapa de planificación.

- **Planificación**: Una vez realizadas las dos etapas anteriores, el proyecto debe presentar un esquema de trabajo, el cual debe de estar bien definido para no cometer errores con los plazos, una vez realizado dicho esquema de trabajo se procede a la etapa de codificación, dicho esquema de trabajo esta presentado en la Figura 3.1 y detallado en cada uno de los puntos presentados a continuación.

- **Codificación**: En esta fase del proyecto ya se conoce de forma general que acciones hay que realizar para obtener los resultados deseados, pero es una de las fases que mas tiempo se toma para asegurar los correctos resultados. En nuestro proyecto, esta fase consiste en realizar una arquitectura basada en Kubernetes[1] (k8s, explicado en la sección 4.2) para gestionar los datos descargados desde la web de empatica[2], siendo estos procesados EDWAR (Merino Semprún, 2020) y así luego almacenarlos en las bases de datos InfluxDB (sección 4.4.2.1) y MariaDB (sección 4.4.1.2) para su posterior uso en la plataforma de gestión para los doctores, todo este proceso esta documentado en el capítulo 5. Una vez terminada esta etapa el proyecto debe ser funcional y se procede a realizar la etapa de pruebas.

- **Pruebas**: En esta etapa, se realizan pruebas unitarias sobre el software para dar una demostración de que el sistema construido realmente funciona, en nuestro proyecto esta fase se implementó de forma manual, donde se han probado cada uno de los apartados que deben de estar presentes dentro de la página web. Esto permite comenzar con la última fase del proceso, la implementación.

- **Despliegue**: En esta última fase del proyecto, se realiza la puesta en producción del proyecto, en nuestro caso llevar el modelo de Kubernetes a una plataforma online como lo es Google Compute Engine[3] (GCE, explicado en la sección 4.10.2), para que sea accesible por los usuarios finales. Además, en esta etapa también se realiza la documentación del proyecto.

[1] `https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/` Accedido: 15/05/2021

[2] `https://www.empatica.com/connect/login.php` Accedido: 15/05/2021

[3] `https://cloud.google.com/compute` Accedido: 15/05/2021

Capítulo 4

Herramientas

Para la implementación de este proyecto, se ha planteado el uso de diferentes programas, frameworks y plataformas los cuales se describen a continuación para facilitar la comprensión de la lectura de la implementación de nuestro proyecto.

4.1 Docker

Docker[1] es una capa de abstracción sobre el sistema operativo que nos permite administrar contenedores de forma rápida y sencilla para que las aplicaciones que allí se ejecutan tengan un ambiente limpio y no compartan información con otras aplicaciones. Un contenedor es código y dependencias (programas, ficheros, entre otros archivos necesarios para el correcto funcionamiento de la aplicación a ejecutar) empaquetadas que permiten ejecutar una aplicación de forma rápida y confiable independientemente del sistema operativo donde se ejecute.

En la Figura 4.1 podemos observar dos tipos de arquitecturas. La primera (situada a la izquierda) utiliza Docker y podemos distinguir cuatro capas, la primera capa es la infraestructura, es decir el hardware que posee el ordenador, la segunda es el sistema operativo de la maquina que tiene el ordenador, la tercera capa es Docker y luego como cuarta tenemos las aplicaciones en los contenedores, en general este esquema se puede resumir diciendo que en las arquitecturas con Docker el sistema operativo es común para todas las aplicaciones, además de que no importa que arquitectura se tenga como núcleo. Por otra parte, la segunda arquitectura (situada a la derecha) utiliza maquinas virtuales y podemos distinguir tres capas, la primera es la infraestructura, la siguiente es un hipervisor[2], que es un software que

[1] `https://www.docker.com/resources/what-container` Accedido: 16/05/2021
[2] `https://es.wikipedia.org/wiki/Hipervisor` Accedido: 16/05/2021

virtualiza hardware para poder ejecutar diferentes sistemas operativos en un mismo ordenador, la tercera son las máquinas virtuales, que cada una se subdivide en dos capas, el sistema operativo de la máquina virtual y además la aplicación que se va a ejecutar en cada una de ellas.

Una vez dicho esto, en nuestro proyecto encaja mejor una arquitectura Docker ya que, al querer llevar a cabo un proyecto en la nube se pueden evitar problemas de ejecución en máquinas virtuales generados por la falta de dependencias.

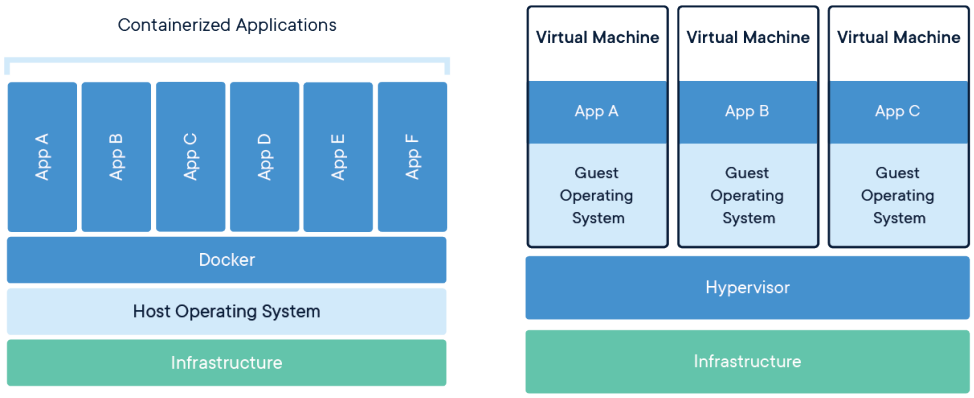

Figura 4.1: Arquitectura de aplicaciones en contenedores y en máquinas virtuales.

Fuente: Docker

4.2 Kubernetes

Kubernetes[3], también conocido como k8s, es una plataforma portable y extensible que permite la administración de cargas de trabajo y servicios, además de que es un entorno que está centrado en contenedores. Un nodo es una máquina que realiza las tareas solicitadas, es decir se encarga de que los *pods* (sección 4.2.1) creados por los *deployments* (sección 4.2.3) y *statefulsets* (sección 4.2.4) sean generados correctamente en un espacio limpio. En la Figura 4.2 podemos observar un nodo de Kubernetes en el cual se están desplegando varios *pods* y dentro cada cada uno de ellos las aplicaciones en contenedores y volúmenes (sección 4.2.2) que necesitan para su correcto funcionamiento.

4.2.1 Pod

Un *pod* es un conjunto de uno o más contenedores de Docker que poseen almacenamiento y red compartidos y además contienen explicaciones de como

[3] `https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes` Accedido: 16/05/2021

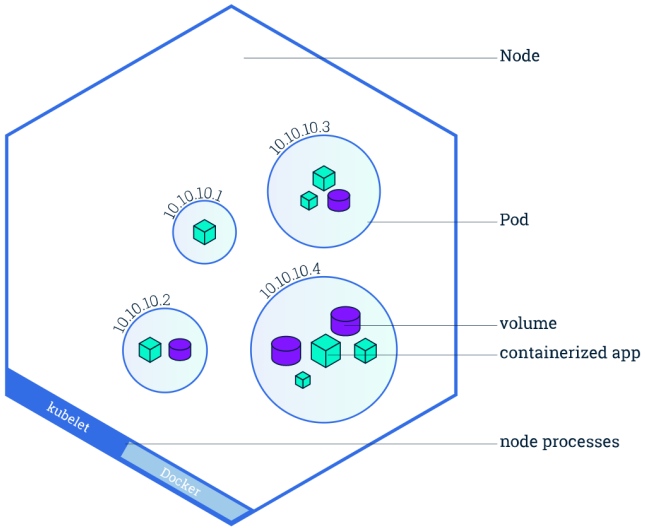

Figura 4.2: Orquestando contenedores con Kubernetes.
Fuente: Kubernetes

ejecutar los contenedores que habitan en él.

4.2.2 Volume

Un volumen es una unidad de almacenamiento de información teniendo en cuenta que estos discos virtuales se crean previamente o cada vez que se genera el *pod* para así poder permitir la compartición de memoria o la permanencia de los datos.

4.2.3 Deployment

Un *deployment*[4] es un controlador que se encarga de proporcionar actualizaciones declarativas y de cambiar el estado de forma controlada de los diferentes recursos que se encuentran en él.

4.2.4 StatefulSet

Un *statefulset*[5] permite gestionar aplicaciones con estado para el despliegue y escalado de un conjunto de *pods*. Se diferencia de un *deployment* ya que garantiza la unicidad y el orden de cada uno de los *pods*. Los identificadores

[4] `https://kubernetes.io/es/docs/concepts/workloads/controllers/deployment` Accedido: 16/05/2021

[5] `https://kubernetes.io/es/docs/concepts/workloads/controllers/statefulset` Accedido: 16/05/2021

de los *pods* en un *statefulset* son persistentes y se mantienen iguales con cualquier actualización de estado.

4.3 Helm

Helm[6], es un manejador de paquetes para Kubernetes, en el cual se crean *charts*[7] (paquetes que contienen las definiciones necesarias para ejecutar una aplicación) para permitir un despliegue más sencillos de aplicaciones complejas dentro de un clúster de Kubernetes. Dichos paquetes se pueden encontrar en los repositorios[7] (lugares donde se almacenan los charts creados para su posterior uso y distribución) que, al instalarse sobre un clúster, generan realeases[7] (instancias de un *chart* en un clúster de Kubernetes) para su identificación dentro del clúster.

En informática, un clúster[8] es un conjunto de ordenadores con los cuales se puede interactuar como si se tratase de un único ordenador.

4.4 Bases de datos

Una base de datos, es un conjunto de datos almacenados en una memoria externa que están organizados mediante una estructura de datos. Existen dos tipos de bases de datos, relacionales y no relacionales, los cuales serán explicados a continuación.

4.4.1 Bases de datos relacionales

Una base de datos relacional[9] es un tipo de base de datos en el que se almacenan datos y se relacionan entre sí. Este tipo de base de datos están basados en el modelo relacional y en ella cada una de las filas de una tabla es un nuevo registro. Existen muchas bases de datos relacionales, entre ellas podemos destacar MySQL (sección 4.4.1.1) y MariaDB (sección 4.4.1.2) las cuales serán explicadas a continuación.

4.4.1.1 MySQL

MySQL[10] es un sistema de gestión de bases de datos relacionales que esta basado en *Structured Query Language* (SQL) y que además posee un modelo cliente-servidor.

[6] `https://helm.sh` Accedido: 16/05/2021

[7] `https://helm.sh/es/docs/intro/using_helm` Accedido: 16/05/2021

[8] `https://es.wikipedia.org/wiki/Clúster` Accedido: 16/05/2021

[9] `https://www.oracle.com/es/database/what-is-a-relational-database/` Accedido: 16/05/2021

[10] `https://www.hostinger.es/tutoriales/que-es-mysql` Accedido: 16/05/2021

4.4.1.2 MariaDB

MariaDB[11] es un sistema de gestión de bases de datos derivado de MySQL, siendo compatibles entre si, pero tiene un objetivo en concreto y es que con MariaDB se puede cambiar un servidor por otro directamente.

4.4.2 Bases de datos no relacionales

Una base de datos no relacional[12] es un sistema de gestión de bases de datos que no usan SQL como lenguaje principal de consultas además, los datos no contienen una estructura fija (como son las tablas del modelo relacional). Es un modelo de base de datos que escala horizontalmente, es decir que puede tener un amplio número de nodos servidores.

En este trabajo, al necesitar manejar información que proviene de dispositivos de IoT, hemos optado por utilizar una base de datos de series de tiempo que es un tipo de base de datos no relacional que esta optimizado para información que contenga horas, fechas o formatos similares. Este tipo de base de datos manejan métricas, eventos o mediciones que estén acotadas por medio de algún dato de intervalo de tiempo. Un ejemplo de base de datos de series de tiempo es InfluxDB el cual será explicado a continuación.

4.4.2.1 InfluxDB

InfluxDB es una base de datos de series de tiempo de código abierto que esta optimizada para el manejo de información que proviene con intervalos de tiempo además, nos proporciona un lenguaje para realizar consultas que es muy similar a SQL.

4.5 EDWAR

EDWAR (Merino Semprún, 2020) es un módulo de procesamiento de datos provenientes de dispositivos de IoT, que tiene una librería de python en el repositorio de PyPI[13], que permite obtener más información a partir de los datos suministrados por los dispositivos, con ayuda de modelos de inteligencia artificial. Cuando se quiere ejecutar este modelo es necesario indicar de donde provienen los datos y donde están los ficheros con dichos datos al método *edwar.Run* para que este posteriormente permita almacenar la información generada en tablas de la librería Pandas[14], guardarlos como CSV.

Para nuestro proyecto, hemos decidido no utilizar la librería de PyPI ya que EDWAR sigue bajo desarrollo por los miembros del equipo, por ese

[11] `https://es.wikipedia.org/wiki/MariaDB` Accedido: 16/05/2021
[12] `https://es.wikipedia.org/wiki/NoSQL` Accedido: 16/05/2021
[13] `https://pypi.org/project/Edward/` Accedido: 17/06/2021
[14] `https://pandas.pydata.org/` Accedido: 17/06/2021

motivo se ha tomado la decisión de trabajar con el repositorio privado de GitHub[15] que contiene los últimos cambios que aún no han sido llevados a PyPI.

En la Figura 4.3 podemos observar como está definida la arquitectura del módulo de procesamiento donde se aprecia que a la entrada se introducen los ficheros con los datos de los dispositivos IoT y la configuración a utilizar. Posteriormente, con estos datos, EDWAR almacena la información de los ficheros en memoria y procede a su envío a cada uno de los modelos que se especifican en el fichero de configuración (estos modelos se denominan *parser* en la imagen). Por último, una vez obtenida la nueva información se almacena nuevamente para luego ser exportada en ficheros CSV o a una base de datos.

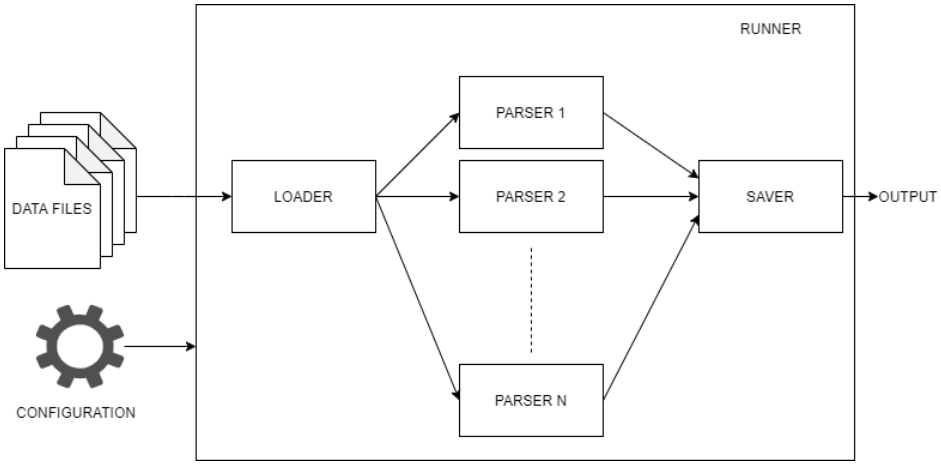

Figura 4.3: Arquitectura del módulo de procesamiento EDWAR.
Fuente: Repositorio EDWAR

4.6 Flask Framework

Flask[16] es un framework del lenguaje de programación Python, que permite crear aplicaciones webs de forma sencilla. Este framework esta siendo utilizado actualmente para construir *Application Programming Interface*[17] (API), software que permite la comunicación entre dos aplicaciones, robustas de una forma rápida, sencilla y eficiente.

[15] `https://github.com/greenlsi/edwar` Accedido: 17/06/2021
[16] `https://flask.palletsprojects.com/en/2.0.x/` Accedido: 17/06/2021
[17] `https://www.mulesoft.com/resources/api/what-is-an-api` Accedido: 17/06/2021

4.7 Grafana

Grafana[18] es un software que permite realizar la visualización de datos, basados en métricas provenientes de distintas fuentes de almacenamiento de información.

En la Figura 4.4, podemos observar el panel de control configurable que ofrece Grafana para visualizar los datos. La información que aquí se muestra es completamente configurable por el usuario y permite mantener control sobre las métricas almacenadas en InfluxDB, además de que también permite tener el control sobre el clúster de Kubernetes para controlar su funcionamiento y consumo.

Figura 4.4: Panel de control de Grafana.
Fuente: Grafana

4.8 Agente servidor

Un Agente servidor es un programa que colecciona métricas de diferentes fuentes para luego distribuirlas a los diferentes servicios que estén subscritos a él. A continuación hablaremos un poco sobre dos de los agentes más conocidos Telegraf (sección 4.8.1) y Kafka (sección 4.8.2).

[18] `https://grafana.com` Accedido: 17/06/2021

4.8.1 Telegraf

Telegraf[19] es una plataforma para recolectar información que admite el uso de plugins como Kubernetes, Kafka, Amazon Web Services, entre otros, para facilitar la lectura de información de diferentes fuentes y también ara agilizar la salida de las métricas.

A continuación, la Figura 4.5 podemos observar un diagrama para el uso de Telegraf junto con InfluxDB, donde si destacamos el módulo de Telegraf, podemos ver que recibe la información desde diferentes sitios, como lo son *clouds* públicos, clouds privados, Centros de datos, etcétera. A continuación Telegraf procesa y envía esa información para su almacenamiento en la base de datos InfluxDB. Por último, este módulo también es capaz de solicitar información a la base de datos para realizar envíos a otras fuentes.

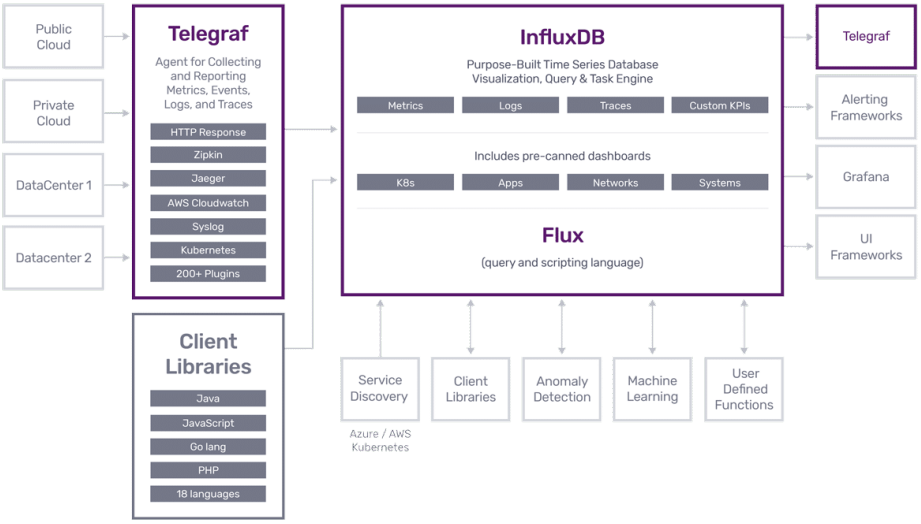

Figura 4.5: Diagrama de uso de Telegraf junto con InfluxDB.
Fuente: InfluxDB

4.8.2 Kafka

Kafka[20] es un proyecto *open-source* que permite la intermediación de mensajes. Es una plataforma que permite la manipulación en tiempo real de diferentes datos. Tiene como ventajas escalabilidad, alto rendimiento, baja latencia y además, permite el manejo de una gran cantidad de datos.

En la Figura 4.6, se aprecia como es el funcionamiento de Kafka. Se observa como recolecta la información desde varios productores, dicha información

[19] `https://www.influxdata.com/time-series-platform/telegraf/` Accedido: 17/06/2021
[20] `https://kafka.apache.org` Accedido: 17/06/2021

es procesada y almacenada en temas. Los consumidores pueden subscribirse a dichos temas, y cuando Kafka almacene información sobre alguno de ellos, enviará la información a cada uno de los consumidores registrados para que pueden hacer uso de ella.

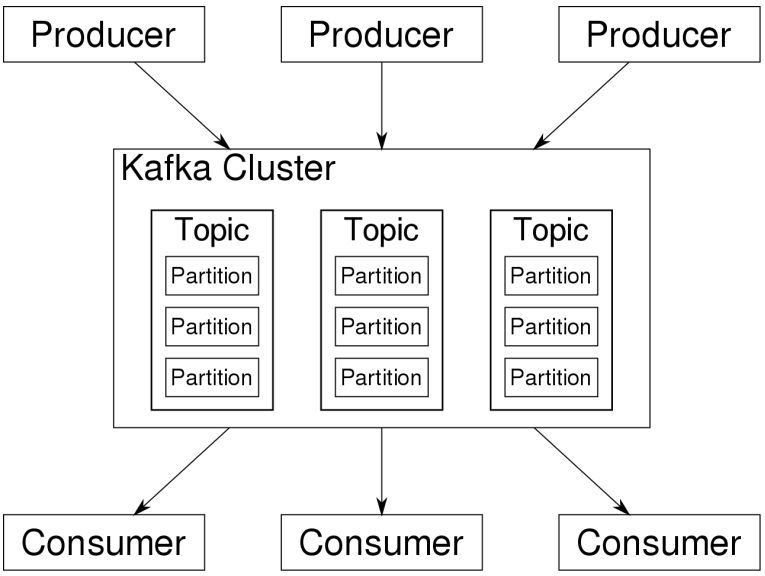

Figura 4.6: Diagrama de uso de Kafka.
Fuente: Kafka

4.9 NGINX

NGINX[21] es un servidor web que es utilizado como proxy inverso[22], que garantiza protección a los dispositivos de una red de cualquier influencia procedente de la web, una cache de HTTP y como un balanceador de carga[23], se encarga de distribuir las llamadas a la aplicación entre los diferentes servidores que se tengan disponibles.

4.10 Cloud Services

Cloud[24] es un conjunto de servidores remotos a los que se pueden acceder en todo momento desde cualquier ubicación con una conexión a internet.

[21] `https://kinsta.com/es/base-de-conocimiento/que-es-nginx/` Accedido: 17/06/2021

[22] `https://www.ionos.es/digitalguide/servidores/know-how/que-es-un-servidor-proxy-inverso/` Accedido: 17/06/2021

[23] `https://es.wikipedia.org/wiki/Equilibrador_de_carga` Accedido: 17/06/2021

[24] `https://www.ionos.es/digitalguide/servidores/know-how/que-es-el-cloud/` Accedido: 17/06/2021

Estos servidores ofrecen *Cloud Services*[25] (servicios en la nube), los cuales son servicios que se ofrecen *on demand* (bajo petición) a empresas y usuarios por medio de una conexión de internet. Están diseñados para dar fácil acceso a aplicaciones y recursos, sin necesidad de adquirir el hardware necesario.

Algunos ejemplos de servicios en la nube son Amazon Web Services (sección 4.10.1) y Google Cloud (sección 4.10.2) de los cuales hablaremos a continuación.

4.10.1 Amazon Web Services

Amazon Web Services[26], es una plataforma en la nube desarrollada por Amazon que ofrece una variedad de servicios a nivel global, entre esos servicios podemos destacar *Amazon Elastic Kubernetes Service* (sección 4.10.1.1) del cual hablaremos a continuación.

4.10.1.1 Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service[27] (EKS), es un gestor de clústeres de Kubernetes, seguros y de alta disponibilidad, en la nube que permite ejecutar y escalar aplicaciones de Kubernetes.

En la Figura 4.7 observamos como es el flujo de trabajo para ejecutar aplicaciones en Kubernetes con *Elastic Kubernetes Service*, se aprecia como el primer paso es crear un clúster de Amazon EKS. A continuación, se inician las maquinas virtuales que contendrán nuestra aplicación de Kubernetes. Una vez el clúster esté preparado, se configuran las herramientas necesarias para el manejo del clúster, como *kubectl*. Por último, se observa la implementación y ejecución de las aplicaciones bajo un entorno de Kubernetes.

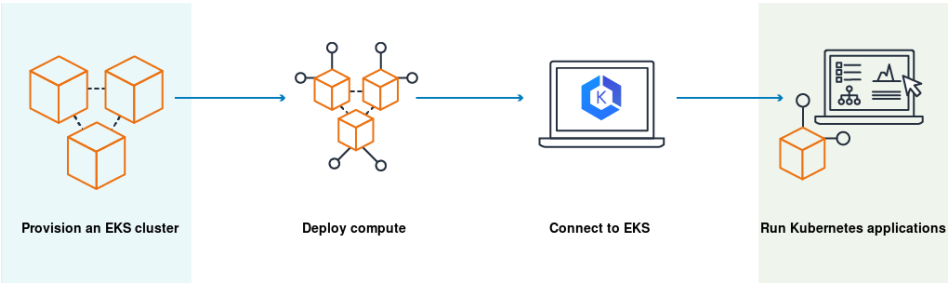

Figura 4.7: *Amazon Elastic Kubernetes Service.*
Fuente: Amazon

[25] `https://www.citrix.com/es-es/solutions/digital-workspace/what-is-a-cloud-service.html` Accedido: 17/06/2021
[26] `https://aws.amazon.com/es/what-is-aws/` Accedido: 17/06/2021
[27] `https://docs.aws.amazon.com/eks/latest/userguide/` Accedido: 17/06/2021

4.10.2 Google Cloud

Google Cloud[28], es una plataforma en la que se gestionan en la nube diferentes servicios ofrecidos por la compañía, siendo para nosotros de especial relevancia *Google Kubernetes Engine* (sección 4.10.2.1) el cual será explicado a continuación.

4.10.2.1 Google Kubernetes Engine

Google Kubernetes Engine[29] (GKE), es un servicio de clústeres de Kubernetes para la implementación, administración y escalabilidad de aplicaciones en contenedores utilizando la infraestructura proporcionada por Google.

En la Figura 4.8 podemos observar la arquitectura de un clúster de *Google Kubernetes Engine*, donde se aprecia que el usuario por medio de la herramienta *kubectl*, que permite la administración del clúster de Kubernetes, realiza diversas acciones en el "plano de control". Dichas acciones crean nodos, con las aplicaciones en contenedores, o administran servicios de Google como por ejemplo, balanceadores de carga, discos, redes, entre otros. Una vez todos los nodos y servicios estén inicializados el usuario puede acceder a ellos para hacer uso de la aplicación implementada.

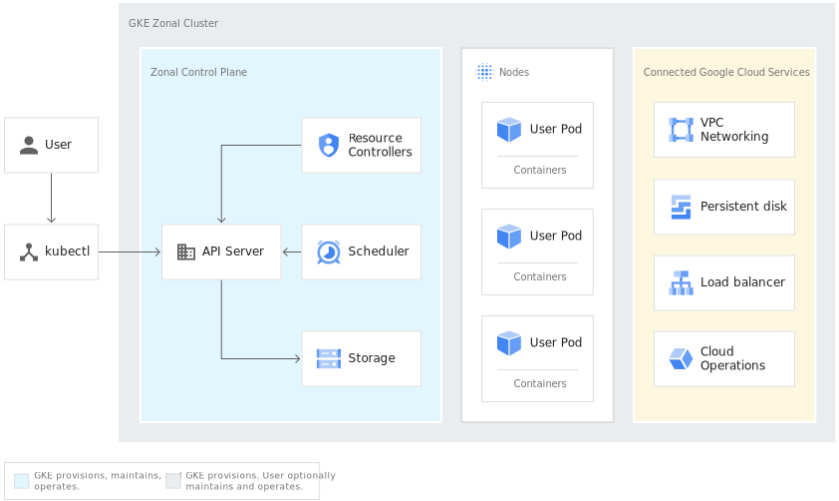

Figura 4.8: *Google Kubernetes Engine.*
Fuente: Google

[28] `https://cloud.google.com/why-google-cloud?hl=es` Accedido: 17/06/2021
[29] `https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview?hl=es-419` Accedido: 17/06/2021

Capítulo 5

Implementación

En este capítulo hablaremos del proceso de implementación del sistema, describiendo detalladamente cada uno de los procesos que hemos realizado para su obtención, pasando por cada una de las etapas que hemos modificado hasta llegar a la implementación final.

5.1 Arquitectura de la solución

La arquitectura de solución, ha sufrido pequeñas variaciones desde el primer diseño hasta el presentado finalmente. A continuación describiremos cada una de las arquitecturas que hemos diseñado en el proceso de creación del clúster de Kubernetes.

5.1.1 Primera arquitectura presentada

Para la realización del presente trabajo fue necesario describir una primera aproximación a la arquitectura de la solución, la cual se puede observar completamente en el diagrama presente en la Figura 5.1. Para una mayor facilidad y entendimiento sobre cada uno de los módulos allí presentes, en las siguientes secciones desglosaremos y explicaremos la función de cada uno de ellos.

5.1.1.1 Módulo de InfluxDB

Como hemos explicado anteriormente, InfluxDB es una base de datos de series de tiempo (sección 4.4.2.1). En la Figura 5.2, se puede observar un *StatefulSet* (sección 4.2.4). Esta configuración hace más fácil el acceder a la información almacenada en los volúmenes creados, los cuales estarían disponibles por medio de las herramientas de *Amazon Web Services*, dando como

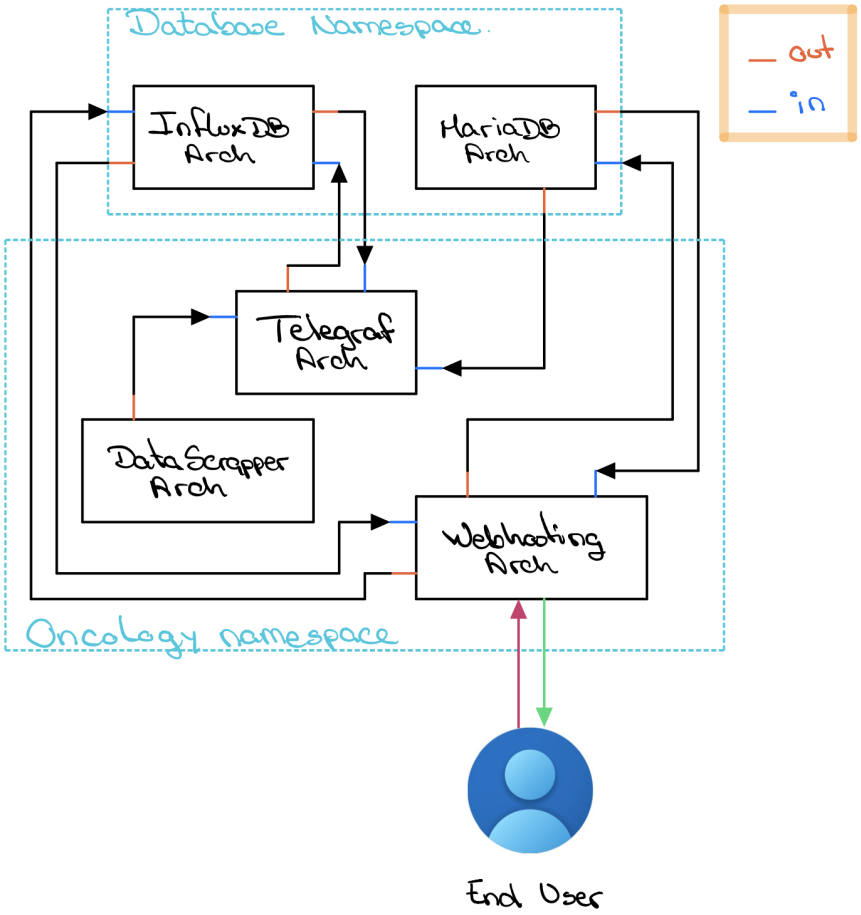

Figura 5.1: Diagrama de la primera arquitectura de la solución.

primera aproximación un *Bucket de S3*[1]. Por otra parte podemos observar el servicio de conexión a este módulo que nos permitirá conexiones privadas provenientes únicamente dentro del clúster.

5.1.1.2 Módulo de MariaDB

Para no sobrecargar la base de datos de InfluxDB, se decidió implementar una base de datos relacional que nos permitirá acceder a los datos de los pacientes y tendrá información para obtener los datos de la base de datos de series temporales, así como información clínica del paciente recogida, por ejemplo mediante registro web o mediante aplicación móvil si existe. Dicha base de datos es MariaDB, la cual se explica en la sección 4.4.1.2.

[1] `https://aws.amazon.com/es/s3` Accedido: 18/06/2021

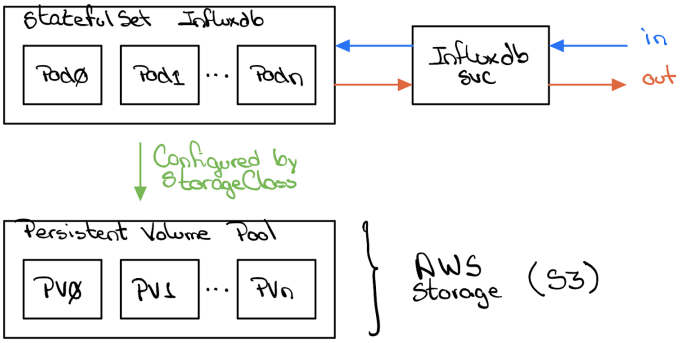

Figura 5.2: Esquema del módulo de InfluxDB.

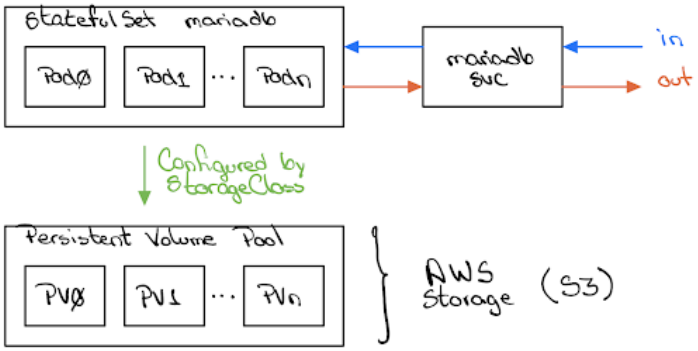

Figura 5.3: Esquema del módulo de MariaDB.

5.1.1.3 Módulo de Telegraf

Como ya hemos explicado anteriormente en la sección 4.8.1, es un agente servidor que permite distribuir la información obtenida.

Este módulo está pensado para recibir los datos que se tengan que guardar en la base de datos de series de tiempo, ya que Telegraf nos permite recoger métricas de los sensores y luego enviarlas a InfluxDB para su almacenamiento. En la Figura 5.4 podemos observar un Deployment con una sola réplica y un servicio, por tanto este módulo en cuanto a la arquitectura de Kubernetes es de los más básicos de este proyecto, junto con el de recopilación de datos (sección 5.1.1.4).

5.1.1.4 Módulo de recolección de datos

Este módulo es esencial para el proyecto, ya que, como se ha indicado anteriormente en otras secciones (sección 1.1 y 2.3) deben extraer los datos desde

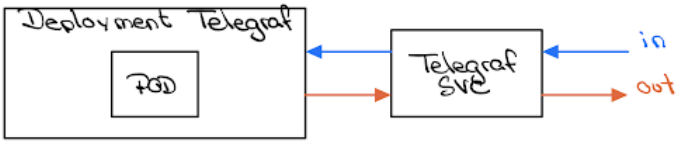

Figura 5.4: Esquema del módulo de Telegraf.

la web de Empatica[2] para luego enviarlos y almacenarlos en la base de datos de series temporales para que sea fácil el acceso por el médico a través del módulo de web hosting.

En la Figura 5.5, podemos observar que este módulo, en cuanto arquitectura, es básico, solo es necesario un *deployment* y un servicio para poder tener una dirección IP dentro del clúster y permitir el envío de datos al módulo de Telegraf.

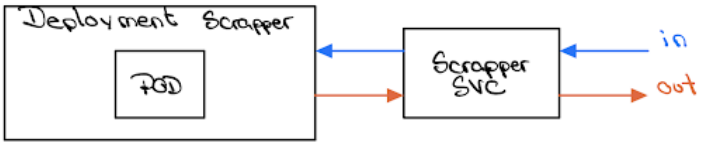

Figura 5.5: Esquema del módulo de recolección de datos.

5.1.1.5 Módulo de web hosting

Por último, tenemos el módulo que gestiona la plataforma web, el cual será el que permitirá la interacción de los médicos con los datos generados por sus pacientes. En la Figura 5.6, podemos observar como este web hosting consta de un *deployment* (sección 4.2.3) que posee varias réplicas para prevenir el fallo en uno de ellos y que permita la conexión por parte del usuario en caso de fallo.

Esta arquitectura ha tenido algunos problemas, y por eso hemos decido realizar pequeños cambios que nos llevaron a cambiar algunas de las cosas, como veremos en la sección 5.1.2.

5.1.2 Segunda arquitectura presentada

Como segunda arquitectura se implementaron cambios menores, para solucionar algunos problemas obtenidos con el desarrollo. El problema principal que derivó en el cambio de arquitectura, es que el módulo de Telegraf (sección 4.8.1) no funcionaba correctamente, por ese motivo hemos decidido cambiarlo por un módulo que también es un agente servidor, Kafka, explicado en

[2] `https://www.empatica.com/en-eu/research/e4/` Accedido: 18/06/2021

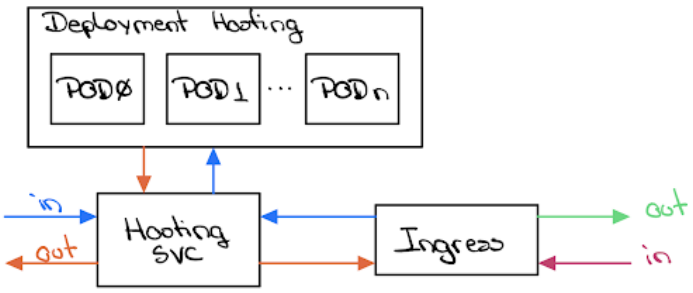

Figura 5.6: Esquema del módulo de *webhosting*.

la sección 4.8.2. Además, otro cambio que se puede observar en esta arquitectura es que hemos eliminado por completo los *namespaces* que teníamos anteriormente debido a que al intentar conectar algunos de los pods, nos daba error. Por otra parte, se nos indico también que los datos descargados debían ser procesados por EDWAR (sección 4.5). En la Figura 5.7, podemos observar el esquema general de esta arquitectura, donde no se observan los *namespaces*, se cambia el módulo de Telegraf y se añaden los módulos de *ingress* y EDWAR.

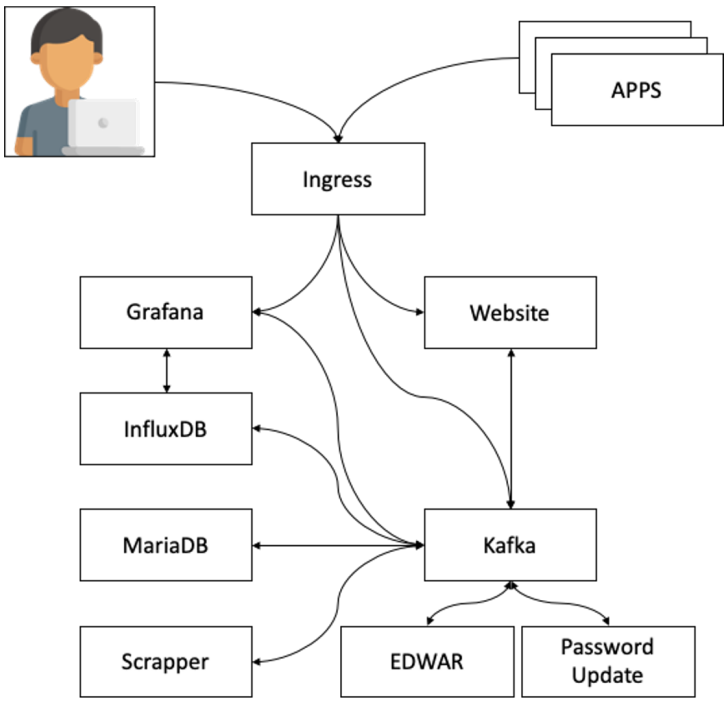

Figura 5.7: Diagrama de la segunda arquitectura de la solución.

5.1.2.1 Módulo de procesamiento de datos

Este módulo, como se explica en la sección 4.5, está pensado par recibir los datos y procesarlos antes de su almacenamiento en la base de datos. En esta primera aproximación hemos intentado utilizar su posibilidad para conectarse directamente a la base de datos, pero lamentablemente no tuvimos éxito en realizar esto.

En la Figura 5.8, podemos observar la conexión que se intento realizar entre los módulos de EDWAR 4.5 e InfluxDB 4.4.2.1, esta conexión no se realizó de esta forma ya que había que realizar modificaciones sobre el código fuente de EDWAR, pero desconocíamos como funcionaba para este aspecto.

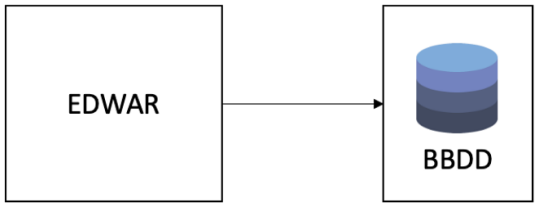

Figura 5.8: Diagrama de la primera conexión intentada para el uso de EDWAR.

5.1.2.2 Módulo de Kafka

Kafka, como explicamos en la sección 4.8.2, es un agente servidor que nos permite recopilar la información y distribuirla entre los diferentes elementos que están suscritos a actualizaciones. Esta funcionalidad es útil para el proyecto ya que se podría visualizar en tiempo real la información suministrada por los dispositivos al mismo tiempo que se almacena en la base de datos.

Se intentó usar este agente servidor ya que teníamos problemas con Telegraf al intentar configurarlo y ya que funcionan de forma similar, se intentó sustituirlo antes de eliminarlo por completo.

Finalmente, este módulo se ha retirado de la arquitectura final ya que no contribuía a mejorar la solución obtenida y solo aportaba un aumento en la complejidad.

5.1.2.3 Módulo de cambio de contraseña

Este módulo aparece en este momento, ya que se detecta una necesidad con la web de Empatica, y es que cada tres meses pide al usuario un cambio de contraseña. Para los médicos es un trabajo tedioso el tener que cambiar la contraseña de los dispositivos y enviarla a los pacientes para que puedan descargar la información.

5.1.2.4 Módulo de ingress

El módulo de *ingress* es básicamente un servicio de NGINX (sección 4.9), el cual nos permite acceder a la pagina web que se encuentre de dentro del clúster de Kubernetes.

Este módulo, para la fase de desarrollo tiene configurado tres URLs que apuntan a una sola IP, pero se distribuyen entre las aplicaciones. En la figura 5.9 podemos observar la configuración de desarrollo para el módulo de ingress, donde se ven configurados los hosts "oncologytfg.es" y "backend.oncologytfg.es", lo cuales apuntan a la dirección IP "172.16.241.18" en el puerto "80".

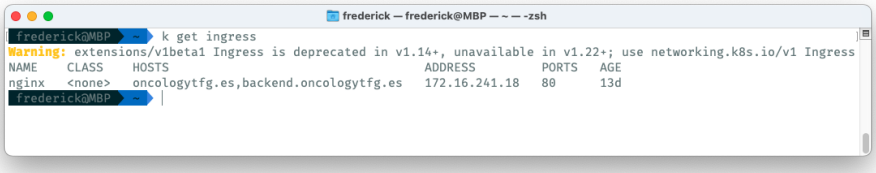

Figura 5.9: Configuración de desarrollo del módulo de ingress.

5.1.3 Arquitectura final

Luego de intentar la arquitectura anterior, nos hemos percatado de pequeños errores que hacían que nuestra aplicación no fuese del todo funcional, razón por la cual surge el modelo descrito en la Figura 5.10, donde se observan pequeñas diferencias como por ejemplo, ya no hay módulos de agente servidor, en su lugar encontramos un Backend que se encarga de gestionar las peticiones y enviarlas a sus respectivos destinos finales. Además, también hemos eliminado el módulo de Grafana (sección 4.7) ya que no aportaba funcionalidad al proyecto final. A continuación explicaremos los cambios que se han realizado en esta nueva arquitectura.

5.1.3.1 Módulo de procesamiento de datos

En la arquitectura anterior, el módulo de procesamiento de datos tenía un problema, el cual no permitía que se procesaran los ficheros debido a que al iniciar el pod con el código, este se ejecutaba y luego se quedaba bloqueado, es decir no recibía la información de forma correcta.

Por este motivo, para obtener un módulo funcional se construyó una capa de abstracción sobre EDWAR (Merino Semprún, 2020), para que esta aplicación se quedara escuchando a peticiones por medio de un API. Una vez que se recibe el fichero comprimido con formato zip, se procede a descomprimirlo

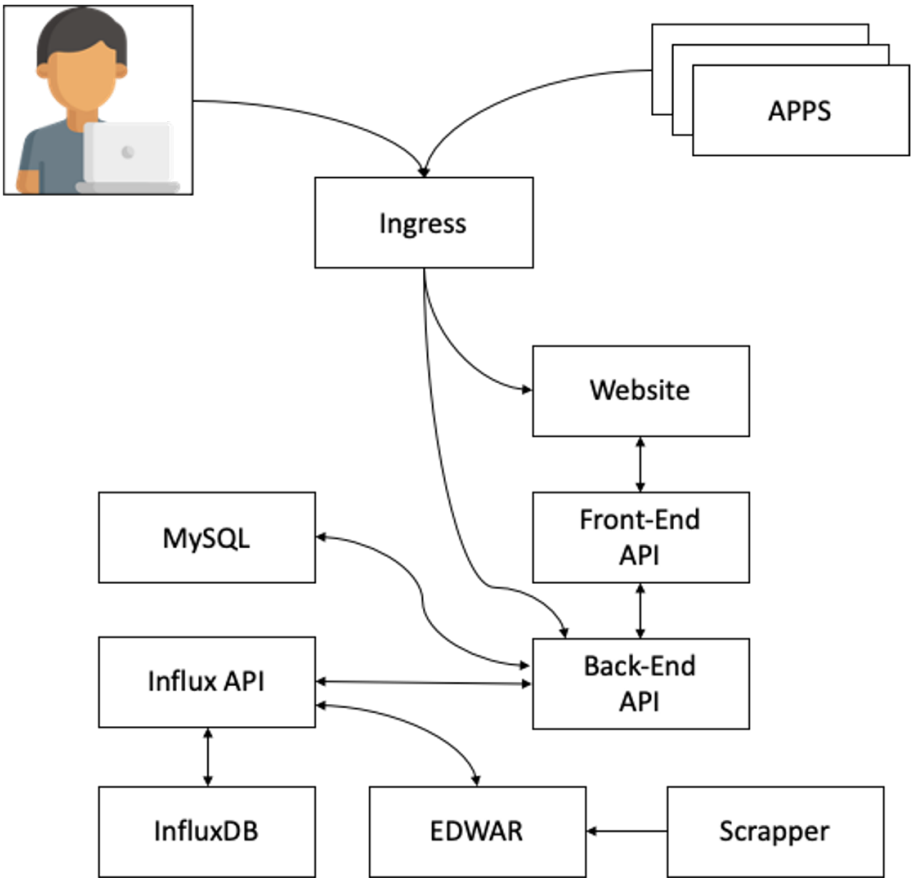

Figura 5.10: Diagrama de la arquitectura final de la solución.

y a procesar la información obtenida por medio de EDWAR.

En la Figura 5.11 podemos observar como el módulo que antes solo era EDWAR ahora lo absorbe un API para escuchar a las peticiones provenientes del Scrapper (sección 5.2), el cual envía los ficheros ZIP con los datos recopilados por las pulseras de monitorización.

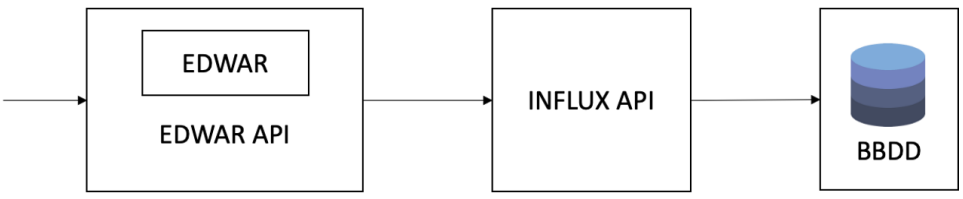

Figura 5.11: Diagrama de la conexión implementada para el uso de EDWAR.

5.1.3.2 Módulo de base de datos de series de tiempo

Anteriormente el BackEnd API se conectaba directamente a la base de datos InfluxDB, pero debido a la gran cantidad de peticiones que genera el módulo de EDWAR, hemos decido desacoplar esta funcionalidad para que no sean lentas las peticiones que necesita el sitio web.

5.1.3.3 Módulo de Backend

Este módulo se crea para sustituir a Kafka y Telegraf. Se hace este cambio ya que, para los requerimientos actuales de la web, no hace falta incluir la complejidad de un Agente Servidor. Este módulo consta, de una API la cual se describe en la sección 5.3.1, siendo el módulo central del sistema, ya que todas las comunicaciones importantes pasan directamente por él.

5.1.3.4 Módulo de Frontend

Este módulo se crea para formalizar la construcción de una aplicación robusta, es el encargado de suministrar las vistas a la pagina web y además, si en algún momento es necesario, hacer llamadas al Backend (sección 5.1.3.3) para obtener la información a mostrar en las vistas o para enviar la información que se debe almacenar. De forma general, podemos describir este módulo como un interconector entre los datos y el usuario final.

En las secciones que se encuentran a continuación, se describe como se han implementado de los módulos que tiene esta arquitectura final.

5.2 Recolección de datos

El módulo de recolección de datos es uno de los más simples dentro de la arquitectura, es un módulo que implementa una aplicación de Python que se descarga las sesiones de los dispositivos directamente desde la página de Empatica E4[3], utilizando la librería de PyPI E4Client[4]. Gracias a este módulo realizamos la descarga de la información de forma periódica (una vez al día) y luego enviará los datos al siguiente módulo que se encargará de procesar los datos.

5.3 Application Programming Interfaces

Para facilitar el trabajo a realizar en este proyecto y obtener comunicación entre cada uno de los pods que se encuentran en el clúster hemos implementado cuatro Application Programming Interfaces (APIs) con la ayuda

[3] `https://www.empatica.com/connect/login.php` Accedido: 18/06/2021
[4] `https://pypi.org/project/e4client/` Accedido: 18/06/2021

de Flask (sección 4.6), para el BackEnd (sección 5.3.1), para el FrontEnd (sección 5.3.2), para la comunicación con la base de datos InfluxDB (sección 5.3.3) y como hemos mencionado anteriormente, otra para EDWAR (sección 5.3.4).

5.3.1 Backend API

Este API consta de rutas para gestionar la información que se encuentra en el sistema. A continuación se detallan cada una de las rutas disponibles:

- **[GET] /register-options**: Devuelve las opciones posibles para todos los desplegables en la vista de registro de paciente.
- **[GET] /register-doctor-options**: Devuelve las opciones posibles para todos los desplegables en la vista de registro de un médico.
- **[GET]/get-wearable-options**: Devuelve las opciones posibles del tipo de wearable.
- **[GET] /patients-paginated**: Devuelve la lista de pacientes activos o inactivos dado un número de página y un número de filas a mostrar.
- **[GET] /doctors-paginated**: Devuelve la lista de médicos dado un número de página y un número de filas a mostrar.
- **[GET] /wearables-paginated**: Devuelve la lista de wearables registrados dado un número de página y número de filas a mostrar.
- **[GET] /get-patient-count**: Devuelve la cantidad de pacientes activos o inactivos registrados en el sistema.
- **[GET] /get-doctors-count**: Devuelve la cantidad de médicos registrados en el sistema.
- **[GET] /get-wearable-count**: Devuelve la cantidad de wearables registrado en el sistema.
- **[GET] /get-patient**: Devuelve la información de un paciente.
- **[GET] /get-doctor**: Devuelve la información de un doctor.
- **[GET] /get-wearable**: Devuleve la información de un wearable.
- **[GET] /get-patients-options-info**: Devuelve la información de las opciones de un paciente, esto debido a que al obtener la información del paciente, algunos valores son solo identificadores y no textos.

- **[GET] /get-doctor-options-info**: Devuelve la información de las opciones de un médico, esto debido a que al obtener la información del médico, algunos valores son solo identificadores y no textos.
- **[GET] /get-wearable-options-info**: Devuelve la información de las opciones de un wearable, esto debido a que al obtener la información del wearable, algunos valores son solo identificadores y no textos.
- **[GET] /empatica-credentials**: Devuelve una lista con todos los usuarios registrados en el sistema que tengan usuario y contraseña de Empatica.
- **[GET] /check-session**: Devuelve si una sesión ya está almacenada en el sistema, o no.
- **[POST] /login**: Verifica el usuario y contraseña suministrado y si coincide con los registros de la base de datos devuelve la información del usuario, en caso de que no coincidan devuelve un mensaje de error que indica que no es un usuario o contraseña válida.
- **[POST] /doctor**: Una vez que se ha hecho login como médico, se llena la información de la pantalla inicial con esta llamada. Es decir, se devuelve cantidad de pacientes activos, cantidad de pacientes inactivos y la cantidad de médicos registrados.
- **[POST] /create-patient**: Recibe la información para registrar un paciente en la base de datos, en caso de ser correcta almacena los datos y devuelve un mensaje de paciente registrado.
- **[POST] /create-doctor**: Recibe la información para registrar un médico en la base de datos, en caso de ser correcta almacena los datos y devuelve un mensaje de médico registrado.
- **[POST] /create-wearable**: Recibe la información para registrar un wearable en la base de datos, en caso de ser correcta almacena los datos y devuelve un mensaje de wearable registrado.
- **[POST] /set-active**: Deshabilita a un usuario (paciente o médico).
- **[POST] /set-inactive**: Habilita a un usuario (paciente o médico).
- **[POST] /update-patient**: Recibe datos actualizados de un paciente y realiza los cambios en la base de datos, al finalizar devuelve un mensaje de paciente actualizado.
- **[POST] /update-doctor**: Recibe datos actualizados de un médico y realiza los cambios en la base de datos, al finalizar devuelve un mensaje de médico actualizado.

- **[POST] /update-wearable**: Recibe los datos actualizados de un wearable y realiza los cambios en la base de datos, al finalizar devuelve un mensaje de wearable actualizado.

5.3.2 Frontend API

Este API consta de rutas para gestionar la plataforma web, la cual cuando recibe peticiones por parte de los usuarios envía o solicita la información al BackEnd API. A continuación se detallan cada una de las rutas disponibles:

- **[GET] /**: Renderiza la pantalla inicial para hacer login en el sistema.
- **[GET] /logout**: Limpia la sesión del usuario que estaba dentro del sistema y procede a mostrar la vista de inicio de sesión.
- **[GET] /login**: Redirige a la ruta de inicio de sesión.
- **[POST] /login**: Verifica que el nombre usuario y contraseña suministrado sea válido y lleva al usuario a la pantalla de inicio corresponda (actualmente solo médicos), en caso de no ser válidos, muestra un error que indica que el usuario o contraseña son inválidos.
- **[GET] /doctor**: Renderiza la pantalla de inicio de los doctores con un pequeño dashboard de información y las posibles acciones que pueden realizar.
- **[GET] /doctor/active-patients**: Renderiza la lista de los pacientes activos en el centro donde trabaja el doctor o en caso de ser administrador en todos los centros.
- **[GET] /doctor/inactive-patients**: Renderiza la lista de pacientes inactivos en el centro donde trabaja el doctor o en caso de ser administrador en todos los centros.
- **[GET] /doctor/doctors-list**: Renderiza la lista de médicos registrados en el sistema la cual solo puede tener acceso aquellos que sean administradores.
- **[GET] /doctor/wearables-list**: Renderiza la lista de wearables regitradis en el sistema a la cual solo puede tener acceso aquellos que sean administradores.
- **[GET] /doctor/view-patient**: Renderiza la vista con la información de un paciente en concreto.
- **[GET] /doctor/view-doctor**: Renderiza la vista con la información de un médico en concreto.

- **[GET] /doctor/view-wearable**: Renderiza la vista con la información de un wearable en concreto.
- **[GET] /doctor/set-inactive**: Envía el paciente a la lista de pacientes inactivos. En caso de deshabilitar un doctor, no se le permitiría hacer login nuevamente.
- **[GET] /doctor/set-active**: Envía el paciente a lista de pacientes activos. En caso de habilitar un doctor, se le permitiría hacer login nuevamente.
- **[GET] /doctor/edit-patient**: Renderiza la vista de edición de un paciente con todos los campos del paciente que se desea editar, en caso de no ser un paciente válido redirige a la pantalla de inicio del médico.
- **[POST] /doctor/edit-patient**: Envía la información al Backend para ser actualizada y se redirige a la pantalla de inicio del médico, además se muestra un mensaje para indicar si el paciente fue actualizado correctamente.
- **[GET] /doctor/edit-doctor**: Renderiza la vista de edición de un médico con todos los campos del médico que se desea editar, en caso de no ser un médico válido redirige a la pantalla de inicio del médico.
- **[POST] /doctor/edit-doctor**: Envía la información al Backend para ser actualizada y se redirige a la pantalla de inicio del médico, además se muestra un mensaje para indicar si el médico fue actualizado correctamente.
- **[GET] /doctor/edit-wearable**: Renderiza la vista de edición de un wearable con todos los campos del wearable que se desea editar, en caso de no ser un wearable válido redirige a la pantalla de inicio del médico.
- **[POST] /doctor/edit-wearable**: Envía la información al BackEnd para ser actualizada y se redirige a la pantalla de inicio del médico, además se muestra un mensaje para indicar si el wearable fue actualizado correctamente.
- **[GET] /doctor/add-patient**: Renderiza la vista de crear un paciente con todos los campos vacíos para proceder a introducir la información.
- **[POST] /doctor/add-patient**: Envía la información al BackEnd para crear el paciente, una vez hecho esto, redirige a la pantalla de inicio donde se puede observar un mensaje que indica que el paciente fue creado.

- **[GET] /doctor/add-doctor**: Renderiza la vista de crear un médico con todos los campos vacíos para proceder a introducir la información.
- **[POST] /doctor/add-doctor**: Envía la información al BackEnd para crear el médico, una vez hecho esto, redirige a la pantalla de inicio donde se puede observar un mensaje que indica que el médico fue creado.
- **[GET] /doctor/add-wearable**: Renderiza la vista de registrar un wearable con todos los campos vacíos para proceder a introducir la información.
- **[POST] /doctor/add-wearable**: Envía la información al BackEnd para registrar el wearable, una vez hecho esto, redirige a la pantalla de inicio donde se puede observar un mensaje que indica que el wearable fue creado.

5.3.3 Influx API

Este API tiene como función recibir la información y enviarla directamente a la base de datos o pedir a la base de datos la información necesaria. Como se ha indicado en la sección 5.1.3.2, se ha desacoplado esta funcionalidad del BackEnd API ya que desde EDWAR se realizan muchas llamadas a este método, por tanto, es más factible que la aplicación tenga problemas de rendimiento si la dejamos en el BackEnd. A continuación se detallan cada una de las rutas disponibles:

- **[POST] /write-point**: Escribe solo un punto en la base de datos de series de tiempo.
- **[POST] /write-point-list**: Escribe muchos puntos creando una lista y luego los envía a influxdb para que los almacene.
- **[GET] /get-data**: Obtiene información la tabla solicitada desde influxdb.
- **[GET] /get-data-mean**: Obtiene la información de la tabla solicitada aplicando la función de media en los valores.

5.3.4 EDWAR API

Este API recibe únicamente el fichero ZIP desde el Scrapper, lo descomprime y posteriormente lo procesa y envía al InfluxAPI. Este módulo tiene como única ruta posible: **[POST] /upload**.

5.4 Bases de Datos

Para almacenar la información proveniente de las aplicaciones externas, los dispositivos wearable y la aplicación web, se han utilizado una base de datos relacional (MySQL, sección 4.4.1.1) y una base de datos de series de tiempo (InfluxDB, sección 4.4.2.1). A continuación describiremos la estructura de como están construidas dichas bases de datos.

5.4.1 MySQL

Este módulo es el único que está íntegramente creado con Helm (sección 4.3) ya que al tener problemas con MariaDB (sección 4.4.1.2) hemos decido buscar una implementación que nos permita utilizar dicho programa. A continuación explicaremos el uso de cada una de las tablas que hemos creado para este proyecto:

- **Tabla WEARABLE_TYPE**: Almacena la información de los tipos de dispositivos wearables que se pueden gestionar.

Campo	Tipo	Descripción
ID	INT	Identificador
TYPE_NAME	VARCHAR(255)	Nombre del tipo de wearable

Tabla 5.1: Tabla WEARABLE_TYPE

- **Tabla WEARABLE**: Almacena los dispositivos registrados en el sistema.

Campo	Tipo	Descripción
ID	VARCHAR(100)	Identificador
WEARABLE_TYPE	INT	Identificador del tipo de wearable
USERNAME	VARCHAR(255)	Usuario
PASSWD	VARCHAR(255)	Contraseña

Tabla 5.2: Tabla WEARABLE

- **Tabla MEDICAL_CENTER**: Almacena los centros médicos disponibles.

Campo	Tipo	Descripción
ID	INT	Identificador
NAME_CENTER	VARCHAR(255)	Nombre del centro médico

Tabla 5.3: Tabla MEDICAL_CENTER

- **Tabla ALCOHOL_OPTION**: Almacena las opciones de bebida de alcohol para los pacientes.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.4: Tabla ALCOHOL_OPTION

- **Tabla DIAGNOSIS_OPTION**: Almacena las opciones de los diagnósticos del paciente.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.5: Tabla DIAGNOSIS_OPTION

- **Tabla FREQUENCY_CRISIS_OPTION**: Almacena la frecuencia de las crisis de un paciente.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.6: Tabla FREQUENCY_CRISIS_OPTION

- **Tabla BODY_PARTS_OPTION**: Almacena la partes del cuerpo a seleccionar.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.7: Tabla BODY_PARTS_OPTION

- **Tabla CARDIOVASCULAR_RISK_OPTION**:Almacena las opciones de riesgo cardiovascular.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.8: Tabla CARDIOVASCULAR_RISK_OPTION

- **Tabla QUALITY_PAIN_OPTION**: Almacena las opciones sobre como es el dolor.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.9: Tabla QUALITY_PAIN_OPTION

- **Tabla SLEEP_DISORDER_OPTION**: Almacena si el paciente tiene problemas para dormir.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.10: Tabla SLEEP_DISORDER_OPTION

- **Tabla TIME_EFFECTIVENESS_MEDICINE_OPTION**: Almacena las opciones para el tiempo de efectividad de la medicación.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.11: Tabla TIME_EFFECTIVENESS_MEDICINE_OPTION

- **Tabla DEFINITION_MOMENTS**: Almacena los momentos en los que se debe tomar la medicación.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción

Tabla 5.12: Tabla DEFINITION_MOMENTS

- **Tabla MEDICINES_OPTION**: Almacena las posibles medicinas que se pueden indicar.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de las medicinas

Tabla 5.13: Tabla MEDICINES_OPTION

Campo	Tipo	Descripción
ID	INT	Identificador
GENDER_NAME	VARCHAR(255)	Nombre de los géneros

Tabla 5.14: Tabla GENDER

- **Tabla GENDER**: Almacena los posibles géneros de las personas.
- **Tabla WORKING_OPTION**: Almacenas las opciones de trabajo para los pacientes.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre de la opción de trabajo

Tabla 5.15: Tabla WORKING_OPTION

- **Tabla WORKING_PLACE**: Almacena el lugar de trabajo de un paciente.

Campo	Tipo	Descripción
ID	INT	Identificador
PLACE_NAME	VARCHAR(255)	Nombre del lugar del trabajo

Tabla 5.16: Tabla WORKING_PLACE

- **Tabla WORKING_SECTOR**: Almacena el sector de trabajo de un paciente.

Campo	Tipo	Descripción
ID	INT	Identificador
SECTOR_NAME	VARCHAR(255)	Nombre del sector del trabajo

Tabla 5.17: Tabla WORKING_SECTOR

- **Tabla STUDY_OPTION**: Almacena los posibles tipos de estudio.

Campo	Tipo	Descripción
ID	INT	Identificador
OPTION_NAME	VARCHAR(255)	Nombre del tipo de estudio

Tabla 5.18: Tabla STUDY_OPTION

- **Tabla USER**: Almacena todo la información referente a los usuarios del sistema, sin distinguir entre pacientes y médicos.

Campo	Tipo	Descripción
ID	INT	Identificador
EMAIL	VARCHAR(255)	Email
PASSWD	VARCHAR(255)	Contraseña
FIRST_NAME	VARCHAR(100)	Nombre
LAST_NAME_ONE	VARCHAR(100)	Primer apellido
LAST_NAME_TWO	VARCHAR(100)	Segundo apellido
ACTIVE	BOOLEAN	Usuario activo
PHONE	VARCHAR(100)	Teléfono
MEDICAL_CENTER_ID	INT	Identificador del centro médico
GENDER_ID	INT	Identificador del tipo de género
STUDY_OPTION_ID	INT	Identificador del tipo de estudio

Tabla 5.19: Tabla USER

- **Tabla DOCTOR**: Almacena toda la información que es únicamente sobre los médicos.

Campo	Tipo	Descripción
ID	INT	Identificador
ADMINISTRATOR	BOOLEAN	Administrador
CREATION_DATE	DATE	Día de creación

Tabla 5.20: Tabla DOCTOR

- **Tabla PATIENT**: Almacena toda la información que es únicamente sobre los pacientes.
- **Tabla PATIENT_WEARABLE**: Almacena los registros de prestamos de dispositivos wearables a un paciente.
- **Tabla EMPATICA_SESSIONS**: Almacena información de las sesiones de Empatica descargadas.
- **Tabla DATES_PATIENT**: Almacena las fechas relevantes para los pacientes, como el inicio y fin de estudio.

5.4.2 InfluxDB

La estructura de InfluxDB, al ser una base de datos de series temporales, no es en forma de tablas, la información simplemente se gestiona en forma de registros que tienen diferentes etiquetas que nos sirven para identificarlos posteriormente. Por ese motivo, para simplificar hemos decido establecer como etiqueta, el id del usuario, el id de wearable con el que fue generado y la

Campo	Tipo	Descripción
ID	INT	Identificador
BIRTHDATE	DATE	Fecha de cumpleaños
SMOKING	BOOLEAN	Es fumador
CAFEIN	BOOLEAN	Consume cafeína
DIET	BOOLEAN	Dieta
IMC	DOUBLE	Valor del IMC
STARTING_AGE	INT	Edad inicio enfermedad
METASTASIS	BOOLEAN	Sufrió metastasis
PAIN_RELIEF_MEDICINES	BOOLEAN	Alivio del dolor con medicinas
PAIN_RELIEF	BOOLEAN	Alivio de dolor sin medicinas
DISTRESS	INT	Valor angustia
NORMAL_LIFE	INT	Valor de dificultad vida normal
EFFECTIVENESS_MEDICINE	INT	Efectividad de la medicina
EXERCISES	BOOLEAN	Realiza ejercicio
HOME_POST_CODE	VARCHAR(20)	Código postal casa
WORK_POST_CODE	VARCHAR(20)	Código postal trabajo
ALCOHOL_OPTION_ID	INT	Identificador de la opción de alcohol
DIAGNOSIS_OPTION_ID	INT	Identificador del diagnostico
FREQUENCY_CRISIS_OPTION_ID	INT	Identificador de frecuencia de crisis
BODY_PARTS_OPTION_ID	INT	Identificador de las partes del cuerpo
CARDIOVASCULAR_RISK_OPTION_ID	INT	Identificador del riesgo cardiovascular
QUALITY_PAIN_OPTION_ID	INT	Identificador de la cualidad del dolor
SLEEP_DISORDER_OPTION_ID	INT	Identificador de los transtornos del sueño
TIME_EFFECTIVENESS_MEDICINE_OPTION_ID	INT	Identificador de tiempo de efectividad de la medicina
PREVENTIVE_MEDICINES_OPTION_ID	INT	Identificador de las medicinas preventivas
DEFINITION_MOMENTS_OPTION_ID	INT	Identificador de la definición de momentos
RESCUE_MEDICATION_OPTION_ID	INT	Identificador de las medicinas de rescate
DOCTOR_ID	INT	Identificador del doctor
WORKING_OPTION_ID	INT	Identificador de la opción de trabajo
WORKING_PLACE_ID	INT	Identificador del lugar de trabajo
WORKING_SECTOR_ID	INT	Identificador del sector de trabajo
COMMENTS	VARCHAR(500)	Identificador de los comentarios

Tabla 5.21: Tabla PATIENT

Campo	Tipo	Descripción
PATIENT_ID	INT	Identificador del paciente
WEARABLE_ID	INT	Identificador del wearable
ASSIGN_DATE	DATE	Fecha de asignación
RETURN_DATE	DATE	Fecha de devolución
EMPATICA_PASSWD	VARCHAR(255)	Contraseña empatica

Tabla 5.22: Tabla PATIENT_WEARABLE

Campo	Tipo	Descripción
SESSION_ID	INT	Identificador de la sesión
PATIENT_ID	INT	Identificador del paciente

Tabla 5.23: Tabla EMPATICA_SESSIONS

medición que se está almacenando. A continuación se detallan las estructuras que se han creado en esta base de datos:

- **Measurement HR**: Contiene los datos procesados que permiten observar el ritmo cardiaco del paciente.

Campo	Tipo	Descripción
USER_ID	INT	Identificador del paciente
STARTING_DATE	DATE	Fecha inicio del estudio
END_DATE	DATE	Fecha fin del estudio

Tabla 5.24: Tabla DATES_PATIENT

- **Measurement TEMP** Contiene los datos de la temperatura del paciente.
- **Measurement EDA-SCR-SCL-SMNA**: que contiene los valores de electrodermal activity (EDA), las componentes tónica o SCL (Skin Conductivity Level), las componentes fasica o SCR (Skin Conductivity Respone) y el actividad nerviosa sudomotora (SMNA).

5.5 Control Pain Portal

Control Pain Portal es el nombre de la aplicación web que se ha desarrollado para gestionar los datos del los dispositivos wearables que se le dan a los pacientes. Al principio nuestros tutores nos han suministrado una web en PHP que ya se tenia hecha para tomarla como referencia, el problema planteado en este punto es que como se había planteado la arquitectura se quería que la web no hiciese llamadas directamente a la base de datos, sino que pasara siempre por el BackEnd creado (sección 5.3.1).

Para esto, hemos decido implementar el sitio web utilizando Flask (sección 4.6), lo cual nos permitió crear una web estable en muy poco tiempo. Para ello hemos tenido que crear uno a uno nuevamente los ficheros HTML de cada una de las vistas presentes en la página web, basándonos en la web suministrada.

El resultado final de esta web puede consultarse en el capítulo 6, en el cual se explica brevemente los resultados obtenidos con nuestro trabajo.

5.6 Puesta en producción

Por último en este capítulo, explicaremos como realizar, paso a paso, la puesta en marcha de la arquitectura:

1. Crear el fichero secreto de Kubernetes con la clave SSH para descargar la información de EDWAR. Esto se realiza mediante la siguiente instrucción:

```
kubectl create secret generic ssh-key
--from-file=id_rsa=<id_rsa path>
```

```
--from-file=id_rsa.pub=<id_rsa.pub path>
```

2. Crear los ficheros con las configuraciones para cada *pod* que lo necesite. Para ello se dejan, solo en los *pod* que necesitan configuración (influxdb y mysql), los ficheros .bck para realizar la configuración necesaria. En estos ficheros se deben escribir los valores solicitados codificados en base64 y luego guardarlos quitándoles la extensión .bck para que posteriormente cuando se utilice el fichero install.sh sea creado correctamente. A continuación se muestra el contenido del fichero secret.yml.bck necesario para la configuración de la base de datos de InfluxDB:

```
apiVersion: v1
kind: Secret
metadata:
  name: influxdb-creds
type: Opaque
data:
  # To encode you can use:
  # echo -n 'your-text-here' | base64
  INFLUXDB_PASSWORD: Base64_Encoded_Text
  INFLUXDB_USERNAME:  Base64_Encoded_Text
  INFLUXDB_DATABASE: Base64_Encoded_Text
  INFLUXDB_HOST: Base64_Encoded_Text
```

3. Si se esta utilizando minikube, se debe habilitar el uso de *ingress*, para ello se debe ejecutar el comando:

```
minikube addons enable ingress
```

4. Luego de esto, tenemos todo el entorno preparado para ejecutar el script de instalación, que se encargará de crear todos los *pods* necesarios para el funcionamiento de la aplicación. A este script se le deben dar permisos de ejecución y posteriormente ejecutarlo:

```
chmod +x install.sh && ./install.sh
```

5. Ahora es el momento de configurar los hosts si se va a utilizar minikube, para ello, debemos obtener la IP que nos habilita *ingress* y crear entradas en el fichero /etc/hosts con los nombres de los hosts que se declaren en la configuración de *ingress*. Esta información la podemos obtener ejecutando el comando:

```
kubectl get ingress
```

Se observaran los hosts que debemos de declarar y la IP que debemos utilizar para configurar el fichero /etc/hosts cuyas nuevas entradas para el caso de la configuración que se aprecia en la Figura 5.9 es:

```
# TFG
172.16.241.18  oncologytfg.es
172.16.241.18  backend.oncologytfg.es
```

6. Una vez hecho esto, procedemos con la configuración de la base de datos. Debemos entrar en el *pod* de mysql y abrir la base datos con los comandos:

```
kubectl exec -it my-cluster-mysql-0 -- /bin/bash

mysql -u <usuario> -p<contraseña>
```

Posteriormente, se deben de crear todos los registros que se encuentran en los ficheros src/sql/create.sql y src/sql/inserts.sql para el correcto funcionamiento de la aplicación.

7. Luego de seguir todos estos pasos, seremos capaces de navegar hasta nuestra pagina web, para comenzar a hacer uso del portal de ayuda implementado.

En los capítulos posteriores se pueden apreciar los resultados obtenidos gracias a esta implementación (capítulo 6). Además también se describe qué aportó cada uno de los integrantes a esta implementación (capítulo 7) y también algunos puntos en los que puede aprovechar esta implementación para dar pie a futuros trabajos (capítulo 8).

Capítulo 6

Resultados obtenidos

En este capítulo se mostrarán los resultados obtenidos tras la versión final de la arquitectura (sección 5.1.3) y el diseño e implementación de la plataforma web (capítulo 5) *Control Pain*. Para la realización de todas las vistas se ha hecho uso de Bootstrap, adaptándolo a nuestras necesidades.

Para ello, a través de un flujo de uso guiaremos al lector a lo largo de las diferentes vistas. Lo haremos desde la perspectiva de los distintos tipos de usuarios: administrador y médico.

6.1 Flujo de uso del administrador

En esta sección se simulará el flujo de uso del portal para el caso de un doctor con permisos de administrador (le llamaremos administrador) que, como veremos tiene más funcionalidades que el resto de usuarios.

Para la simulación, se supondrá que el usuario ya existe en la base de datos, es decir, que dispone de un correo y contraseña.

6.1.1 Login

El login, es la página que permite iniciar sesión a cualquier tipo de usuario además, aparecerá cada vez que se intente acceder a la plataforma. Para ello, será necesario proporcionar un correo y contraseña válidos.

En la Figura 6.1, a la derecha, podemos ver como es inicialmente la pantalla de login. A la izquierda, el caso en el que las credenciales son erróneas, por lo que se alerta al usuario que el usuario o contraseña que ha proporcionado son incorrectos. Si por el contrario, son correctos, podrá acceder a la página de *Inicio*.

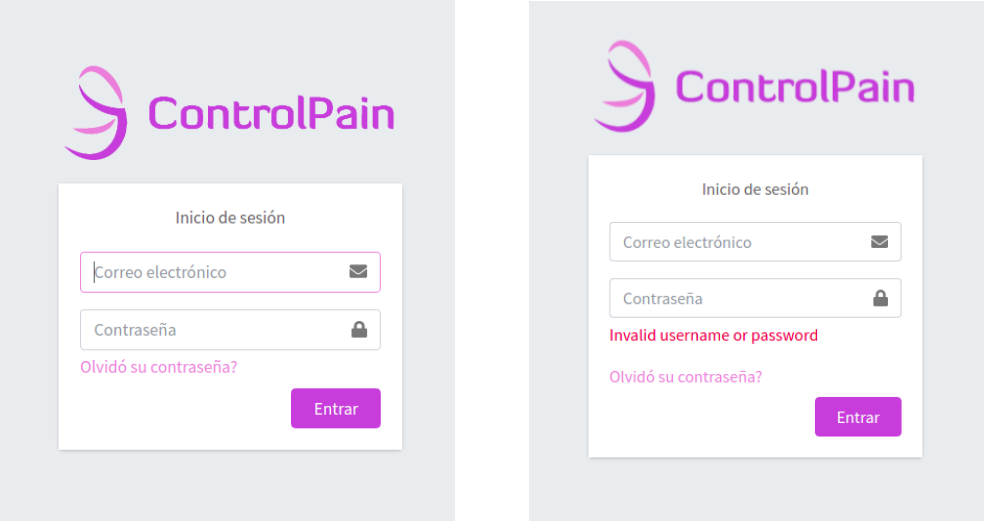

Figura 6.1: Pantalla de login

6.1.2 Inicio

Una vez se haya iniciado sesión correctamente, se redirigirá al administrador a la pantalla de *Inicio*, como se puede ver en la Figura 6.2. A través de esta, podremos acceder a las diferentes vistas en función de tipo de cada usuario.

Figura 6.2: Pantalla de inicio administrador

En la parte superior, a través de las cuatro líneas horizontales podremos expandir o contraer la barra lateral, a la derecha, al pulsar sobre el icono de usuario para que en este caso, el administrador, pueda acceder a su perfil o cerrar sesión.

A la izquierda, a través de la barra lateral, podremos acceder a las vistas de: *Inicio* (sección 6.2), *Registro Paciente* (sección 6.1.4), *Lista de Pacientes activos e inactivos* (sección 6.1.5), *Registrar médico* (sección 6.1.8), *Listado de Médicos* (sección 6.1.8), *Registrar wearable* (sección 6.1.11) y *Lista de*

wearables (sección 6.1.12).

Por último, en el centro, se puede ver en forma de *widgets*, los datos más importantes: número de pacientes activos e inactivos, médicos registrados y el número de wearables registrados en el sistema. Pulsando en la parte inferior de cada uno de ellos, se puede acceder a las vistas de *Lista de pacientes activos* (sección 6.1.5), *Lista de pacientes inactivos* (sección 6.1.5), *Lista de médicos* (sección 6.1.9), *Registrar wearable* (sección 6.1.11) y *Lista de wearables* (sección 6.1.12) que se explicarán más adelante. Además, cada uno de los *widgets* que aparecen en las diferentes vistas, se pueden contraer pulsando sobre el símbolo que aparece a la izquierda.

6.1.3 Perfil del médico

En la Figura 6.3, se muestra el perfil del administrador. Está formado por un módulo llamado datos personales, en el que se puede ver de manera resumida la información personal del administrador: nombre, primer apellido, segundo apellido, sexo, correo electrónico, teléfono, centro médico, estudio, administrador (si se trata de un médico con permisos de administrador marcará SI) y fecha de creación.

Figura 6.3: Perfil del Médico

Una vez visualizado su perfil, el administrador puede comenzar a hacer uso de algunas de sus funcionalidades. Comenzaremos con el *Registro de pacientes*, al que se puede acceder desde la barra lateral o desde el *Inicio* pulsando en la parte inferior de los *widgets*.

6.1.4 Registro de pacientes

El registro de pacientes está formado por seis módulos, datos personales, datos del trabajo, datos clínicos, hábitos de vida, cuestionario de inclusión y medicación.

En datos personales (Figura 6.4), se rellenará con la información necesaria para identificar al usuario, como el nombre, primer apellido, segundo apellido (opcional), fecha de nacimiento, sexo, correo electrónico, teléfono y código postal de residencia habitual.

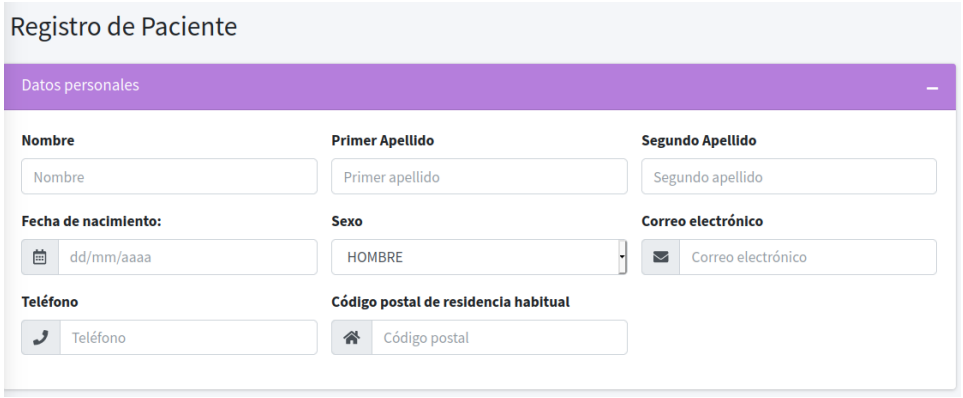

Figura 6.4: Registro de pacientes: datos personales

En el siguiente módulo, datos de trabajo (Figura 6.5), podremos conocer la situación laboral del paciente. En caso de que este trabajando, el código postal de su trabajo, lugar de profesión o la actividad profesional que realiza.

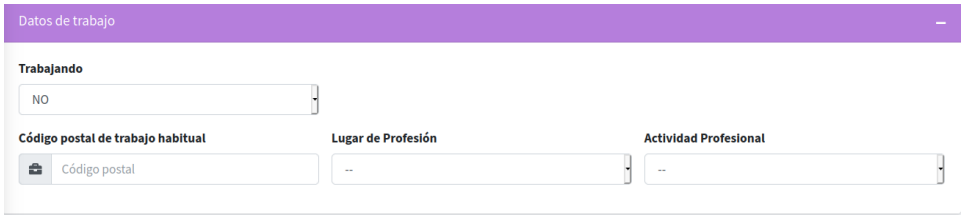

Figura 6.5: Registro de pacientes: datos de trabajo

En cuanto a los datos clínicos (Figura 6.6), se seleccionará el centro médico y estudio asignados al paciente. Dentro de este mismo módulo, se le permitirá al administrador escribir, en el campo de comentarios, cualquier tipo de información de los pacientes que él considere importante.

Una vez se hayan completado los módulos mencionados anteriormente, es importante saber como se encuentra el paciente, tanto física como mentalmente. Por este motivo, en el apartado de hábitos de vida (Figura 6.7), podremos saber si fuma, si consume alcohol o café, frutas y verduras o si realiza ejercicio con frecuencia.

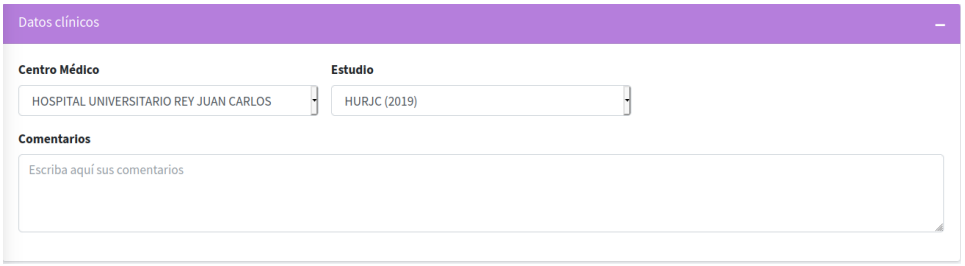

Figura 6.6: Registro de pacientes: datos clínicos

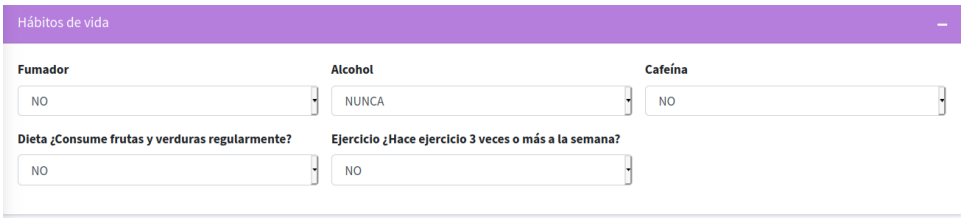

Figura 6.7: Registro de pacientes: hábitos de vida

A través del cuestionario de inclusión (Figura 6.8), tendremos conocimiento acerca de su diagnóstico, si sufre crisis frecuentemente y la edad a la que se inició, metástasis o el caso de que padezca alguna patología como alteración del sueño.

Figura 6.8: Registro de pacientes: cuestionario de inclusión

Finalmente, en el módulo de la medicación (Figura 6.9), se le harán preguntas relacionadas con el dolor que sufre y por último, la medicación que suele tomar.

En cada uno de estos apartados, se incluyen campos con selectores, para que el administrador pueda seleccionar fácilmente entre una variedad de opciones. Además, algunos campos como correo electrónico, teléfono, código postal de residencia o trabajo incluyen iconos, lo que ayuda a su visualización.

Si el administrador ha completado el formulario correctamente, se le redirigirá a la pantalla de *Inicio*, desde la cual, se puede ver que el número de pacientes activos se ha incrementado en uno. En caso contrario, si faltan por

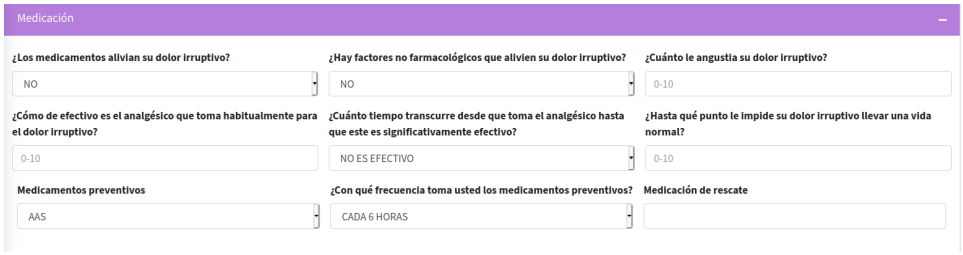

Figura 6.9: Registro de pacientes: medicación

completar algunos campos obligatorios, se le alertará al usuario y no podrá finalizar y enviar hasta que los rellene correctamente.

Desde la pantalla de *Inicio*, el usuario se puede dirigir a la *Lista de pacientes*, como explicaremos en la siguiente sección.

6.1.5 Lista de pacientes

La lista de pacientes, está dividida entre pacientes activos e inactivos. Como se puede ver en la Figura 6.10 y Figura 6.11, son muy similares en cuanto a su estructura y diseño.

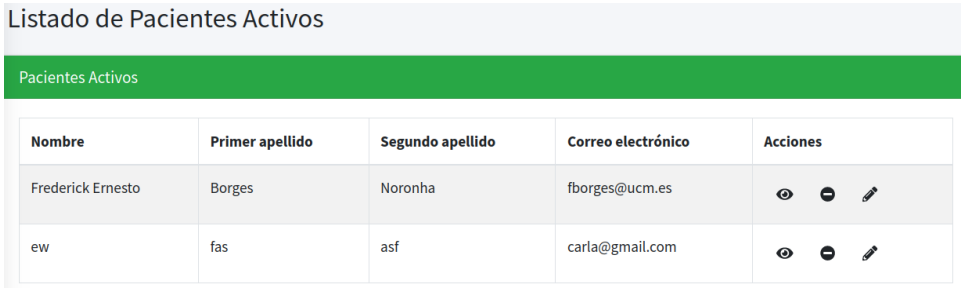

Figura 6.10: Lista de pacientes activos

Figura 6.11: Lista de pacientes inactivos

Cada lista incluye en forma de filas, a los pacientes que actualmente se

encuentran en estudio (activos) o los que lo han finalizado (inactivos). Cada fila está compuesta por el nombre, primer y segundo apellido, correo electrónico y finalmente, un conjunto de acciones. Estas últimas, incluyen unos iconos que permiten acceder de igual manera tanto en la Lista de pacientes activos e inactivos a la pantalla de *Perfil del paciente* (sección 6.1.6) y *Editar paciente* (sección 6.1.7). La única diferencia, se encuentra en la funcionalidad de *Inhabilitar paciente* propia de la *Lista de pacientes activos* y *Habilitar paciente* de la *Lista de pacientes inactivos*. Estas dos funcionalidades no cuentan con una interfaz, sino que simplemente realizan la acción de eliminar un paciente de la lista de activos para añadirlo a la lista de inactivos y viceversa.

A continuación, se explicará la pantalla de *Perfil del paciente* accesible únicamente a través de la lista de paciente activos e inactivos pulsando en el primer icono de la columna acciones.

6.1.6 Perfil del paciente

El *Perfil del paciente*, permite observar toda la información recogida a través del formulario de registro. está dividida en tres módulos, datos personales, datos médicos y datos Empatica E4.

Los dos primeros módulos, datos personales (Figura 6.12), y datos clínicos (Figura 6.13), son iguales a los que hemos explicado en Registro de paciente (sección 6.1.4), con la diferencia de que no son campos que se puedan modificar, son exclusivamente informativos.

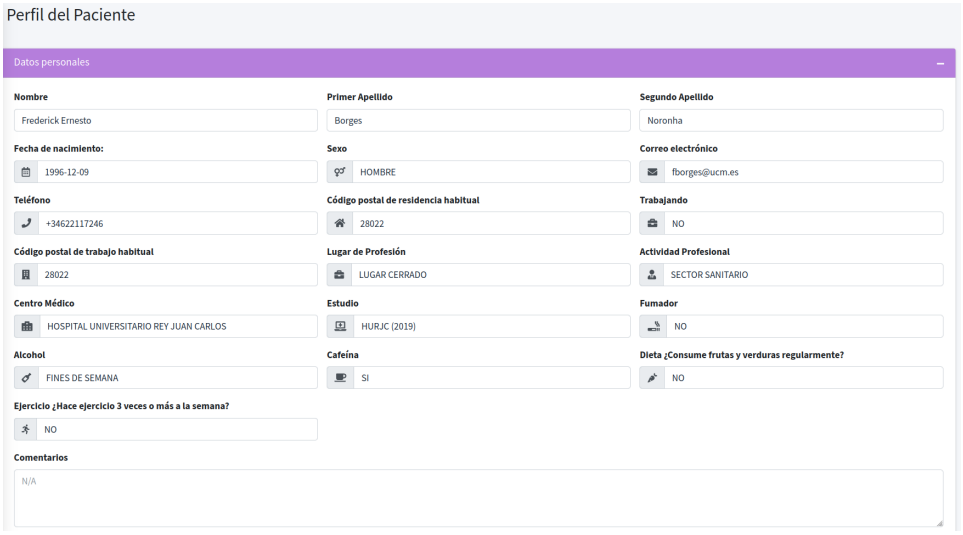

Figura 6.12: Perfil del paciente: datos personales

El último bloque (Figura 6.14), está formado por los datos que se han extraído y procesado de la pulsera E4 tal y como se ha visto en la Implemen-

Figura 6.13: Perfil del paciente: datos clínicos

tación (capítulo 5). Se puede filtrar por día inicio y fin y, posteriormente, pulsando en el botón de Datos empatica, podremos acceder a la Pantalla de gráficas. Esta última, nos permitirá visualizar toda la información relacionada con los datos fisiológicos del paciente.

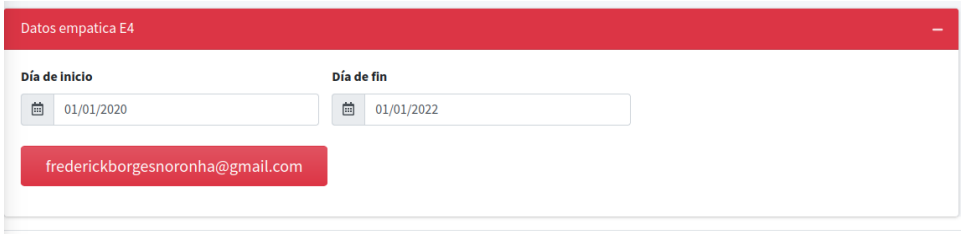

Figura 6.14: Perfil del paciente: datos empatica E4

La vista de gráficas del paciente está formada por tres gráficas donde se muestra la temperatura media, ritmo cardiaco medio por horas y una gráfica de EDA, SCR, SCL Y SMNA medias por hora (Figura 6.15)

La principal finalidad de los tres bloques, se centra en conseguir facilitar al médico el acceso a toda la información personal y fisiológica del paciente de un solo vistazo que, permitirá al médico realizar un seguimiento y evaluación en tiempo real para su posterior diagnóstico.

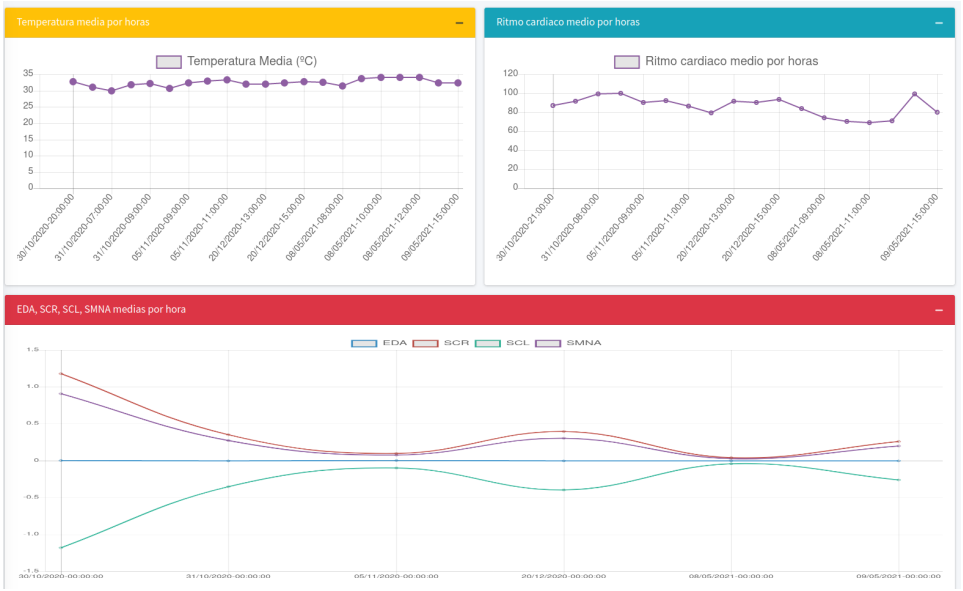

Figura 6.15: Perfil del paciente: gráficas del paciente

Toda la información de está pantalla, salvo los datos de Empatica, se pueden modificar desde la vista *Editar paciente*, como veremos en la siguiente sección.

6.1.7 Editar paciente

Editar paciente, es igual a la pantalla de Registro de paciente (sección 6.1.4), con la diferencia de que aquí se incluyen los datos en cada campo y solo se realiza una modificación en la base de datos mientras que en la de registro, se insertan datos nuevos. Permite editar los datos personales (Figura 6.16), de trabajo (Figura 6.17), clínicos (Figura 6.18), hábitos de vida (Figura 6.19), sobre el cuestionario de inclusión (Figura 6.20), y medicación (Figura 6.21).

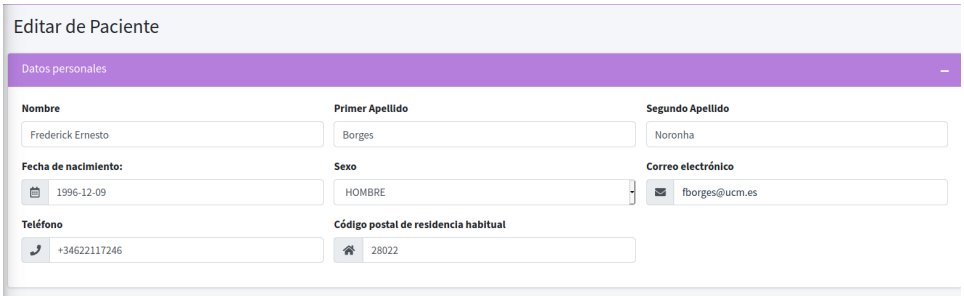

Figura 6.16: Editar paciente: datos personales

Como ya hemos comentado anteriormente, el administrador cuenta, ade-

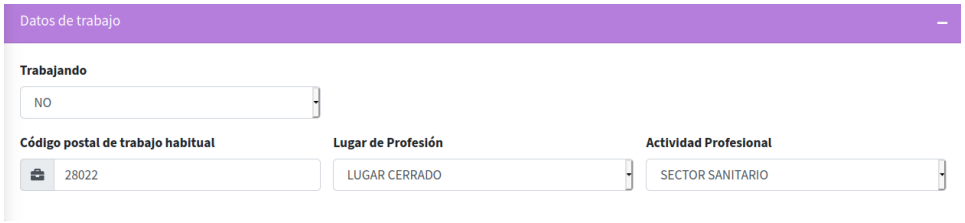

Figura 6.17: Editar paciente: datos de trabajo

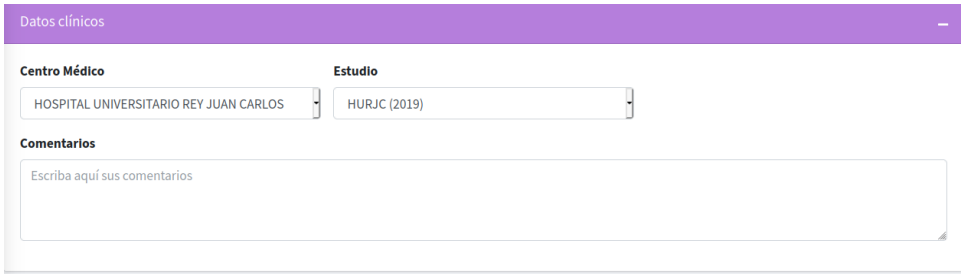

Figura 6.18: Editar paciente: datos clínicos

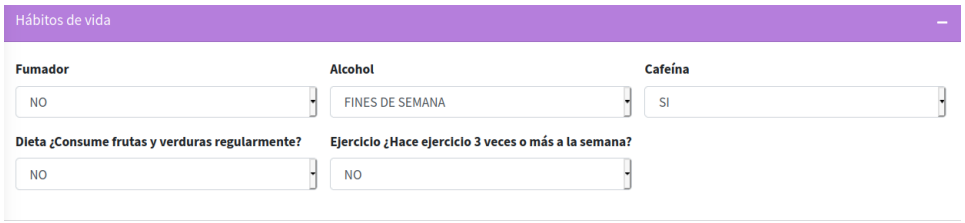

Figura 6.19: Editar paciente: hábitos de vida

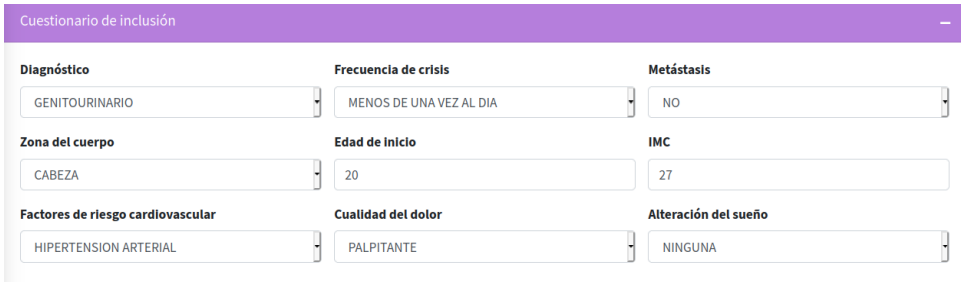

Figura 6.20: Editar paciente: Cuestionario de inclusión

más de las funcionalidades relacionadas con los pacientes, las que permiten la gestión de médicos. Como veremos a continuación, este tipo de usuario puede registrar a otros médicos.

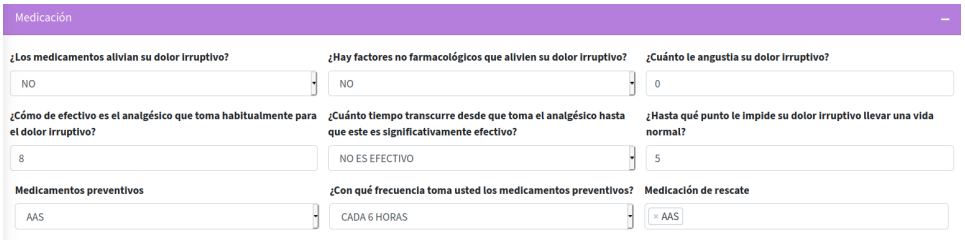

Figura 6.21: Editar paciente: medicación

6.1.8 Registro médicos

Desde la vista de *Registro de médico*, un administrador puede registrar médicos con el rol de administrador o, simplemente, médico.

Como se puede ver en la Figura 6.22, el cuestionario de inclusión está formado por dos bloques, datos personales y clínicos. Los primeros, incluyen campos como nombre, primer y segundo apellido, sexo, correo electrónico y teléfono. El segundo, el rol asignado, centro médico y estudio al que pertenece.

Figura 6.22: Registro de médico

Finalmente, cuando se hayan completado todos los campos obligatorios, se pueden observar a los médicos registrados a través del *Listado de médicos*, presentado a continuación.

6.1.9 Listado de médicos

La estructura y diseño de esta vista, es la misma que la presentada en *Lista de pacientes* (sección 6.1.5), como vemos en la Figura 6.23. Desde esta pantalla, se puede visualizar a los médicos registrados junto con el nombre, apellidos y correo. Además, al igual que para los pacientes, salvo por la función habilitar

y deshabilitar que no está incluida para los médicos, también cuenta con iconos desde los que se puede acceder a *Perfil del médico* (sección 6.1.3, como ya hemos visto anteriormente (sección 6.1.3) y a *Editar médico* como presentaremos a continuación.

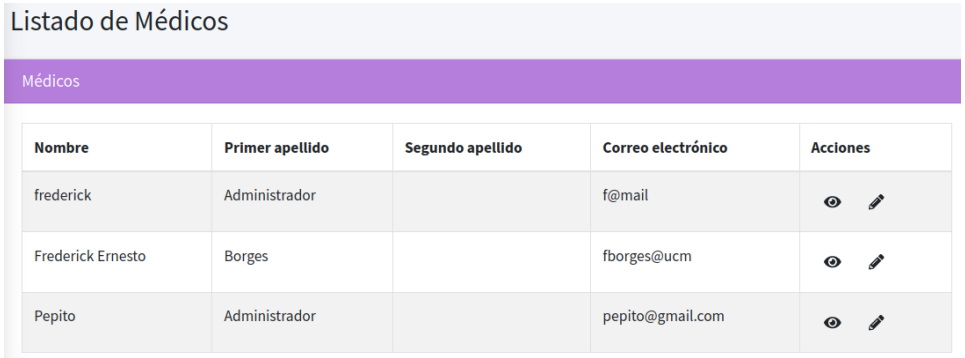

Figura 6.23: Listado de médicos

6.1.10 Editar médico

Como se puede ver en la Figura 6.24, a través de esta vista, se puede editar la información que se había recogido del formulario de registro de médicos (sección 6.1.8). Al igual que para el caso de los pacientes, solo se realizan modificaciones en la base de datos.

Figura 6.24: Editar médico

6.1.11 Registrar wearable

Todas las funcionalidades que están relacionadas con los *wearables* son exclusivas para los administradores.

Como podemos ver en la Figura 6.25, los campos que se deben completar son el ID del *wearable*, tipo de dispositivo (en nuestro caso solo se ha utilizado la pulsera E4), usuario y la contraseña con los que se ha registrado en empatica E4. El tipo de *wearable* se trata de un campo selector, ya que será de mucha utilidad para cuando se incluyan dispositivos diferentes.

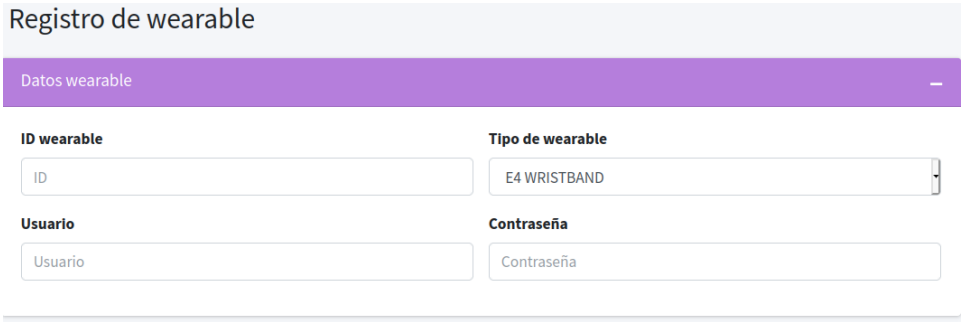

Figura 6.25: Registro de wearable

6.1.12 Lista de wearables

Una vez se hayan registrado los dispositivos wearables, se podrán consultar a través de esta vista (Figura 6.26). Está dividida en forma de lista, donde las columnas son ID, usuario y acciones. Pulsando en los iconos del campo acciones, se podrá acceder a los datos de registro y edición del dispositivo.

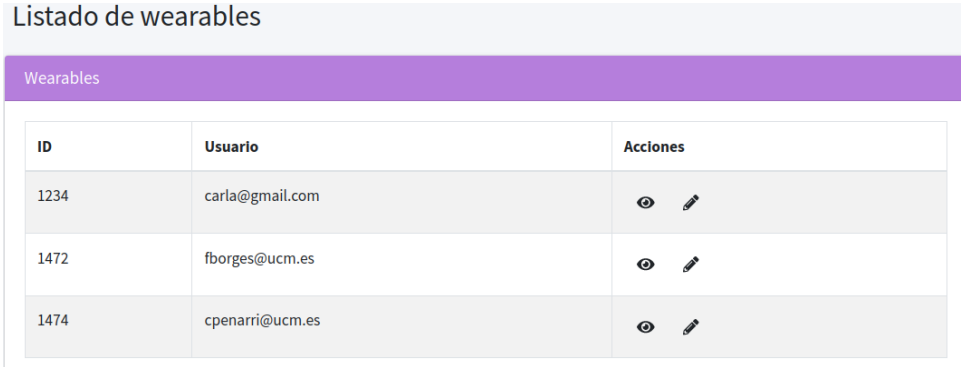

Figura 6.26: Listado de wearables

6.1.13 Datos del wearable

El administrador podrá ver los datos de registro de los dispositivos, de modo que puedan ser consultados en cualquier momento (Figura 6.27).

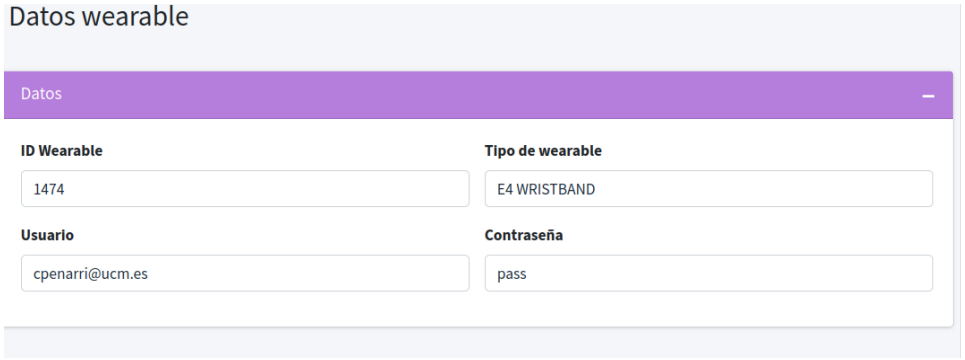

Figura 6.27: Datos del wearable

6.1.14 Editar wearable

Se trata de la misma interfaz de *Registro de wearable* (sección 6.25), con la diferencia de que a través de esta vista se le permite editar en caso de cambiar alguno de los campos (Figura 6.28)

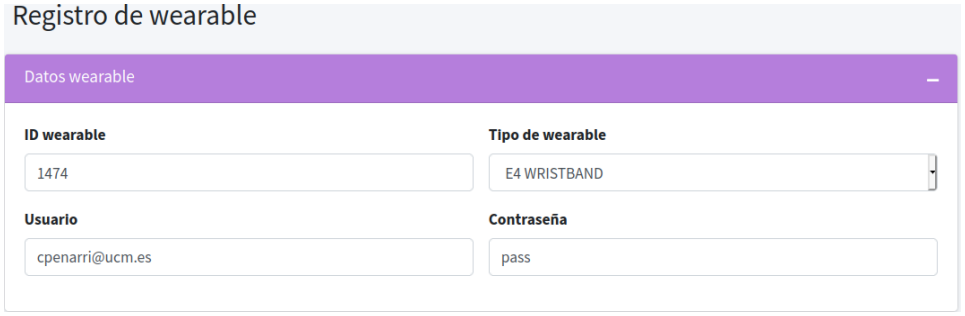

Figura 6.28: Editar wearable

Para terminar, a continuación se explicará el flujo de eso por parte del médico que, como veremos, es muy similar a la del médico administrador.

6.2 Flujo de uso del médico

En esta sección, podremos conocer todas las vistas y funcionalidades a las que tiene acceso el médico. A diferencia del médico administrador, este solo tiene acceso a la información que está relacionada con los pacientes.

6.2.1 Login

El *login*, es el mismo que el explicado en la sección 6.1.1. Una vez que las credenciales sean correctas, podrá acceder a la página de *Inicio*.

6.2.2 Inicio

La pantalla de *Inicio*, es a la que el usuario es dirigido una vez que haya iniciado sesión correctamente. Es muy similar a la del Administrador, con la diferencia de que, el Médico no puede acceder a las vistas de *Registrar médico* (sección 6.1.8) y *Lista de Médicos* (sección 6.1.9), por lo que carece de las funcionalidades que incluyen dichas vistas.

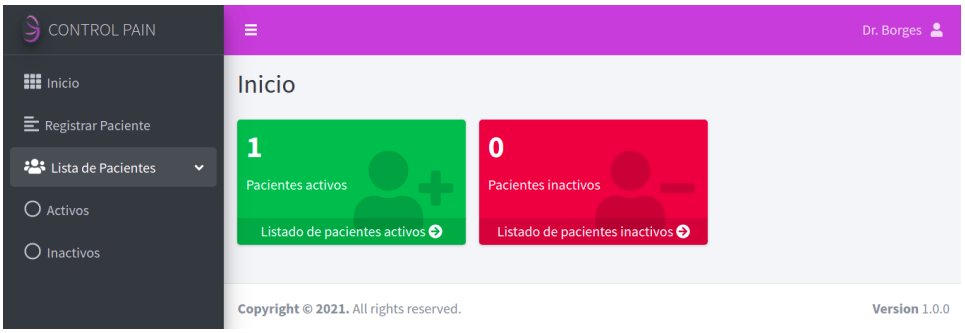

Figura 6.29: Pantalla de inicio médico

En la parte superior, podemos expandir y contraer la barra lateral pulsando sobre las cuatro lineas horizontales y si pulsamos sobre el icono de usuario, ver el perfil del médico y cerrar sesión.

A la derecha, a través de la barra lateral, podemos acceder a la pantalla de *Inicio*, *Registrar Paciente* y *Lista de Pacientes* (activos e inactivos).

En el centro, a modo de "widgets", podemos ver el número de pacientes activos e inactivos y pulsando en la parte inferior de cada uno de ellos, se puede acceder a la *Lista de pacientes activos* y *Lista de pacientes inactivos*.

Una vez el médico haya visualizado la pantalla de *Inicio*, puede acceder a su perfil pulsando en la parte superior de la pantalla.

6.2.3 Perfil del médico

La vista es la misma que la que presentamos en el caso del perfil del administrador (sección 6.1.3), con la excepción de que el campo administrador en este caso marcaría un NO.

Desde la barra lateral, el médico puede comenzar a registrar pacientes como visualizaremos a continuación.

6.2.4 Registro de pacientes

El formulario de *Registro de pacientes*, es idéntico al del flujo de uso del administrador (sección 6.1.4).

Una vez que el paciente haya sido registrado correctamente, se puede acceder a la *Lista de pacientes*.

6.2.5 Lista de pacientes

Esta vista, al igual que la explicada anteriormente (sección 6.1.5), muestra a los pacientes activos e inactivos, respectivamente. A diferencia del médico administrador, el cual tiene acceso a la lista de todos los pacientes registrados, este solo tiene acceso a los que han sido creados por él.

Desde los iconos situados en el campo de acciones, se puede habilitar y deshabilitar a los pacientes y también acceder al *Perfil del Paciente* y *Editar paciente*.

6.2.6 Perfil del paciente

El perfil del paciente es el mismo que el visto previamente *Perfil del paciente* (sección 6.1.6).

Después de que el médico haya estudiado la información del paciente, puede que necesite editarlo en caso de cambios durante el estudio, como se presenta en la siguiente sección.

6.2.7 Editar paciente

La edición del paciente no tiene cambios respecto a lo que habíamos visto en el caso del administrador (sección 6.1.7). Al igual que la lista y el perfil de cada paciente, solo puede ser visualizado por el médico que lo ha registrado o por el administrador.

6.3 Flujo de uso paciente

El paciente solo podrá consultar sus datos personales, ya que nuestra plataforma se centra sobretodo, en las funcionalidades que tiene un médico.

6.3.1 Login

Al igual que hemos visto al principio (sección 6.1.1), el login es el mismo para todos los usuarios. En el caso de los pacientes, supondremos que accede con un usuario y contraseña que ya está en la BBDD.

6.3.2 Perfil Paciente

El perfil del paciente es el mismo que el explicado en la sección 6.1.6. La única diferencia es que el paciente no tendrá acceso a las gráficas que muestran los datos obtenidos de la pulsera biométrica, pues es exclusivo de los médicos.

Capítulo 7

Trabajo individual

En este capítulo describiremos el trabajo individual que ha realizado cada uno de los integrantes de este trabajo, estando descritos detalladamente bajo sus nombres para así poder identificar quien ha realizado cada una de las partes explicadas anteriormente.

7.1 Frederick Ernesto Borges Noronha

Lo primero que tuve que hacer para el proyecto fue investigar los tipos de metodologías, sus ventajas y desventajas, en que situaciones se usa cada una de ellas y cual sería la que mejor se ajuste con nuestro trabajo. Luego escribí acerca de ella en la memoria (capítulo 3), para que mi compañera y yo supiéramos que hacer y que flujo de trabajo se iba a seguir.

Una vez seleccionada la metodología y obtenidos los requerimientos principales de nuestros clientes (en nuestro caso nuestros tutores) empezamos la parte de investigación.

En esta parte, he investigado algunas tecnologías que hemos utilizado en el proyecto: Kubernetes (sección 4.2), Helm (sección 4.3), Docker (sección 4.1), NGINX (sección 4.9) y Grafana (sección 4.7).

Para las bases de datos, al principio nuestros tutores nos aconsejaron utilizar MariaDB (sección 4.4.1.2) e InfluxDB (sección 4.4.2.1), en la cual he participado en su investigación. Cuando he creado el pod con MariaDB fallaba y no permitía su funcionamiento, por ese motivo, he buscado una alternativa, y tras investigar un poco, he seleccionado MySQL (sección 4.4.1.1) ya que funcionaba mejor con Kubernetes.

También he sido responsable de realizar el diagrama de la arquitectura del software. El primero que hice, explicado en la sección 5.1.1, usaba Telegraf (sección 4.8.1), posteriormente lo hemos cambiado por Kafka (sección 4.8.2) en la segunda aproximación (sección 5.1.2). En la última versión del diagrama

(sección 5.1.3), hemos quitado completamente Kafka, y además cambiado la base de datos relacional de MariaDB a MySQL.

Posteriormente, para la implementación del proyecto he realizado varias tareas. He participado en la creación el pod con la instalación de EDWAR, haciendo que este, en una primera fase, descargue el código más reciente desde el repositorio de GitHub, luego ese código se mueve a la carpeta de trabajo donde esta el API que permite su integración con los demás pods, por ultimo se inicia su ejecución para que dicho API espere por envíos del Scrapper para procesar los datos.

También he participado en la modificación del Scrapper para que, en lugar de ser un código que está escrito en una sola función, sea una clase con sus métodos bien especificados y así facilitar su uso.

He realizado también la primera versión del la base de datos en MySQL, siendo esta una versión funcional pero no completa, porque faltaban algunos datos de registro de los pacientes que son importantes para el proyecto.

Para las API que hay en todo el proyecto, he desarrollado la primera versión de cada una de ellas para tener una versión base sobre la cual poder trabajar. Dichas API fueron implementadas con el framework Flask (sección 4.6), que como hemos dicho anteriormente, permite la creación rápida y sencilla de este tipo de aplicaciones.

En el caso del FrontEnd API he desarrollado llamadas para obtener y enviar la información necesaria para cada una de las vistas disponibles en el sistema, esto implica que el FrontEnd API hace llamadas al BackEnd API que es el único que puede realizar consultas en las bases de datos para así tener todo centralizado. El Backend API se encarga de que los datos que se van a almacenar, o consultar sean siempre correctos, por tanto en ella hay consultas para todos la mayoría de los pods que están en el clúster, siempre que requieran hacer uso de las base de datos.

Análogamente, se debía buscar la manera de realizar una comunicación efectiva entre cada uno de los pods de InfluxDB y EDWAR, por este motivo se decidió que la mejor opción era crear para cada una de ellas un API que permitiera esperar peticiones para luego interactuar con los recursos. Por lo tanto, para he implementado ademas del FrontEnd y BackEnd API, dos API adicionales que permiten esperar a una petición para almacenar u obtener los datos de la base de datos de series de tiempo InfluxDB y la segunda permite esperar el envió de información a EDWAR para que directamente empiece con el procesado de los datos que se han obtenido de los dispositivos.

De acuerdo con lo que hemos planificado en nuestra metodología del software, luego de finalizar la implementación de nuestro proyecto, hemos empezado a escribir la documentación. En ella he escrito todo lo que he investigado, además de la documentación sobre EDWAR (sección 4.5), también sobre el Scrapper (sección 2.3). Además, también he escrito el capítulo de Herramientas (capítulo 4), donde explico cada una de las herramientas

necesarias o que fue planteada su utilización y que no fueron creadas por nosotros. Luego, escribí sobre la implementación (capítulo 5), donde se detallan las tres arquitecturas por la hemos pasado. Y para finalizar, escribí las conclusiones (capítulo 9) a las que hemos llegado tras la realización de este trabajo.

7.2 Carla Paola Peñarrieta Uribe

Para poder visualizar la problemática que envolvía nuestro proyecto, tuve que investigar la razón por la cuál es tan importante poder extraer, procesar y analizar los datos provenientes de los diferentes dispositivos weareables. Además, pude focalizar el principal problema que tenemos que abordar con este TFG. Por último, identifiqué los principales objetivos que queremos cubrir con el desarrollo de este proyecto.

Una vez identificado cual es el principal problema, empecé a indagar sobre las posibles soluciones que ya se habían propuesto o desarrollado para intentar resolverlo. Para ello, leí numerosos artículos científicos e investigaciones en los que se proponían posibles soluciones, en muchos casos con resultados prometedores. De esta manera, hemos podido extraer ideas sobre las diferentes propuestas que se habían hecho y analizando si podíamos utilizarlas en nuestro TFG.

Inmediatamente después de conocer la metodología y el flujo de trabajo a seguir, participé en la investigación de Kubernetes (sección 4.2) junto con mi compañero. Al principio de manera más general, con una primera toma de contacto, buscando información en la documentación oficial de la web de k8s y después indagando sobre su implementación.

En cuanto al almacenamientos de los datos, colaboré en el estudio y en cómo se podía proceder a la integración en Kubernetes de las bases de datos recomendadas por los tutores de este proyecto: MariaDB (sección 4.4.1.2) e InfluxDB (sección 4.4.2.1). También investigué sobre el Scrapper (sección 2.3) y EDWAR (sección 4.5) que fue proporcionado por nuestros tutores por lo que tuvimos que entender como estaban implementados. De este último, analicé los distintos archivos que nos proporcionaba tras su ejecución para saber como representar los datos en las correspondientes gráficas que incluiríamos en la vista del médico.

Una vez hecha la investigación se comenzó la fase de implementación. En la que colaboré en la modificación de algunas funciones del Scrapper, adaptándolo según a nuestras necesidades. Implementé una primera versión del pod MariaDB para su posterior integración en Kubernetes. También realicé una primera aproximación de la estructura de la base de datos en MariaDB, analizando que tablas y atributos nos harían falta para el desarrollo de la página web. Posteriormente, escribí en SQL la creación de las tablas en su correspondiente fichero. Desafortunadamente, el pod no funcionaba bien en

Kubernetes por lo que no se utilizó. A pesar de eso, la estructura de la base de datos sirvió como base para la creación de las tablas en MySQL (sección 4.4.1.1).

En lo que se refiere a la web, implementé los HTML de las diferentes vistas, incluido la estructura en JSON para la representación de las gráficas. Participé en la modificación de la base de datos con la creación de nuevas tablas con sus correspondientes atributos. Al realizar dicha modificación, tuve que realizar cambios en los HTML y en la lógica de los mismos, añadiendo nuevas consultas a MySQL. En relación al FrontEnd API y BackEnd API, añadí nuevas funciones como consecuencia de la creación de nuevas tablas. Estas modificaciones permiten que el BackEnd envíe los datos con la nueva información al FrontEnd, al igual que lo hacía en la primera versión de los mismos.

Finalmente, comencé escribiendo en la memoria la Introducción (capítulo 1) y los objetivos (sección 1.1) que ya habíamos visualizado previamente pero que habían ido cambiando a lo largo de su desarrollo. Empezando desde lo más general a lo más particular, hasta llegar a nuestro caso. Seguidamente, en el Estado del Arte (capítulo 2), continué la investigación que había comenzado antes de empezar con el desarrollo del proyecto. Para ello, empecé desde sectores diferentes al que nosotros tenemos que tratar (industria sanitaria) para que el lector pudiese ubicar en que punto o estado se encuentra actualmente el problema a solventar y a continuación, en la industria sanitaria. Por último, en el capítulo de Resultados (capítulo 6), expliqué el flujo de uso de la plataforma web desde la perspectiva cada usuario.

Capítulo 8

Trabajo futuro

En este capítulo hablaremos del trabajo futuro de este proyecto, ya que gracias a nuestra implementación se abren nuevas ventanas en la se permiten a futuros proyectos adaptarlo a sus necesidades con mayor facilidad. A continuación detallaremos una lista de funcionalidades que pueden ser implementadas:

- Permitirá la replicar fácilmente esta arquitectura para otros trabajos del equipo de investigación.
- Utilización de más algoritmos de procesamiento de datos para obtener diferentes resultados, incluso para diferentes estudios que no estén relacionados con oncología.
- Permitir el uso de más dispositivos *wearables*, por ejemplo, dispositivos ambientales, para extraer mayor cantidad de información y así permitir que el médico posea mayor cantidad de información de sus pacientes.
- Inclusión de nuevos módulos, como por ejemplo, un módulo de Big Data y ML. Gracias a eso, se pueden obtener predicciones, que en nuestro caso, podrían prevenir al médico de posibles enfermedades del paciente o posibles crisis.
- Inclusión de un sistema de alertas, en el que en caso de observar anomalías en las gráficas o si existe un módulo de predicciones, se puede alertar al médico o paciente.
- Realizar un sistema de mensajería en el que permita mantener contacto directo al médico y paciente.

Con base en lo anterior hay muchos factores que se pueden mejorar en un futuro en esta aplicación, pero confiamos en que el trabajo actual ha servido de base para la elaboración de futuros desarrollos sobre esta plataforma.

Capítulo 9

Conclusiones

Con este Trabajo Final de Grado hemos conseguido aprender sobre las aplicaciones de la tecnología en el área de la salud, aportando a su vez los conocimientos obtenidos durante nuestros estudios universitarios.

Hemos podido colaborar en la creación de una herramienta, que creemos, será muy útil para cualquier estudio de otras enfermedades, no solo oncológicas y, además, ayudará a mejorar algunos estudios médicos que, son tediosos o que requieren que un doctor siempre esté en contacto con el paciente.

Para la realización de este proyecto debemos indicar que, nuestros estudios como Ingenieros en Informática fueron esenciales para la creación de la plataforma y la interconexión con los diferentes módulos que tiene.

Además, al desarrollarla, hemos dejado cabida para que si se desean agregar más funcionalidades se puede realizar. Todo esto ha sido posible gracias a asignaturas vistas como Base de Datos, Bases de Datos NoSQL, Aplicaciones Web, Fundamentos de la Programación, Tecnología de la Programación y Cloud y Big Data que extendieron el panorama de la programación que teníamos para así finalizar estos años de estudio con este proyecto final.

Para finalizar, de este proyecto hemos aprendido a desplegar una arquitectura completa, en la que se tiene una aplicación ejecutándose y que se va a llevar a un ambiente donde hay un problema real que se quiere solucionar.

Chapter 9

Conclusions

With this Final Project we have managed to learn about the health area technologies, while applying the knowledge we have obtained during our university studies.

We have been able to collaborate in the creation of a tool, which we believe, will be very useful for any study of other diseases, not only oncological and, in addition, will help to improve some medical studies that are tedious or require a doctor to be always in contact with the patient.

For the realization of this project, we must indicate that our studies as Computer Engineers were essential for the creation of the platform and the interconnection with the different modules it has.

In addition to making it, we have left the opportunity to add more functionalities if needed. To do this, we have used the knowledge gained from subjects such as Databases, NoSQL databases, Web Applications, Fundamentals of Programming, Computer Programming Technology and Cloud and Big Data, which extended the programming panorama that we had in order to finish these years of study with this final project.

During this project we have learned to implement and deploy a project with a complex architecture, in which we have a running application and we are going to take it to an environment where there is a real world problem that has to be solved.

Bibliografía

Beg, M., Gupta, A., Stewart, T. y Rethorst, C. Promise of wearable physical activity monitors in oncology practice. *Journal of oncology practice*, vol. 13,2, página 82–89, 2017.

Cappon, G., Acciaroli, G., Vettoretti, M., Facchinetti, A. y Sparacino, G. Wearable continuous glucose monitoring sensors: A revolution in diabetes treatment. *Electronics*, vol. 6(3), 2017. ISSN 2079-9292.

Chaniotis, I. K., Kyriakou, K.-I. D. y Tselikas, N. D. Is node.js a viable option for building modern web applications? a performance evaluation study. vol. 97(10), 2015. ISSN 0010-485X.

Channa, A., Popescu, N. y Rehman Malik, N. Managing covid-19 global pandemic with high-tech consumer wearables: A comprehensive review. En *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, páginas 222–228. 2020.

Durán-Vega, L., Santana, P., Buenrostro, R., Contreras Castillo, J., Anido-Rifón, L., Garcia-Ruiz, M., Montesinos-López, O. y Estrada, F. An iot system for remote health monitoring in elderly adults through a wearable device and mobile application. *Geriatrics*, vol. 4, 2019.

Gao, W., Brooks, G. A. y Klonoff, D. C. Wearable physiological systems and technologies for metabolic monitoring. *Journal of Applied Physiology*, vol. 124(3), páginas 548–556, 2018.

Jamili oskouei, D. R., MousaviLou, Z., Bakhtiari, Z. y Bux Jalbani, K. Iot-based healthcare support system for alzheimer's patients. *Wireless Communications and Mobile Computing*, vol. 2020, páginas 1–15, 2020.

KVEDAR, J., COYE, M. J. y EVERETT, W. Connected health: A review of technologies and strategies to improve patient care with telemedicine and telehealth. *Health Affairs*, vol. 33(2), páginas 194–199, 2014.

LARSON, N. y PINSKER, J. Continuous glucose monitoring in children with type 1 diabetes. *International journal of pediatric endocrinology*, vol. 2013, página 8, 2013.

MANGADO, N. G. y NIETO, M. J. R. Prueba de la marcha de los 6 minutos. *CIBERES*, páginas 15–22, 2016.

MEHTA, D., DESHMUKH, T., SUNDARESAN, Y. B. y KUMARESAN, P. Continuous monitoring and detection of epileptic seizures using wearable device. En *Smart Innovations in Communication and Computational Sciences*, páginas 77–84. Springer Singapore, Singapore, 2019.

MERINO SEMPRÚN, M. Diseño e implementación de una herramienta multiplataforma para extracción de información de dispositivos biométricos iot. *Universidad Politécnica de Madrid*, 2020.

PAGÁN, J., DE ORBE, M. I., GAGO, A., SOBRADO, M., RISCO-MARTÍN, J. L., MORA, J. V., MOYA, J. M. y AYALA, J. L. Robust and accurate modeling approaches for migraine per-patient prediction from ambulatory data. *Sensors*, vol. 15(7), páginas 15419–15442, 2015. ISSN 1424-8220.

PRESSMAN, R. *Ingeniería del software. Un enfoque práctico*. McGraw-Hill, Mexico, D.F., 2010.

ROBLYER, D. M. Perspective on the increasing role of optical wearables and remote patient monitoring in the COVID-19 era and beyond. *Journal of Biomedical Optics*, vol. 25(10), páginas 1 – 9, 2020.

SANNINO, G., FALCO, I. y DE PIETRO, G. Monitoring obstructive sleep apnea by means of a real-time mobile system based on the automatic extraction of sets of rules through differential evolution. *Journal of biomedical informatics*, vol. 49, 2014.

SINGHAL, A. y COWIE, M. The role of wearables in heart failure. *Current heart failure reports*, vol. 17,4, páginas 125–132, 2020.

WHELAN, M. E., ORME, M. W., KINGSNORTH, A. P., SHERAR, L. B., DENTON, F. L. y ESLIGER, D. W. Examining the use of glucose and physical activity self-monitoring technologies in individuals at moderate to high risk of developing type 2 diabetes: Randomized trial. *JMIR Mhealth Uhealth*, vol. 7(10), 2019.

WOOTTON, R. Telemedicine. *BMJ*, vol. 323(7312), páginas 557–560, 2001. ISSN 0959-8138.

Yach, D., Hawkes, C., Gould, C. L. y Hofman, K. J. The global burden of chronic diseasesovercoming impediments to prevention and control. *JAMA*, vol. 291(21), páginas 2616–2622, 2004. ISSN 0098-7484.

Yang, Z., Zhou, Q., Lei, L., Zheng, K. y Xiang, W. An iot-cloud based wearable ecg monitoring system for smart healthcare. *Journal of medical systems*, vol. 40(12), página 286, 2016. ISSN 0148-5598.

Yoon, J. G., Fares, M., Hoyt, W. J. y Snyder, C. Diagnostic accuracy and safety of confirm rx™ insertable cardiac monitor in pediatric patients. *Pediatric Cardiology*, vol. 42, páginas 142–147, 2020.